SIMULIA

3DEXPERIENCE

ABAQUS 2026 FD04
ELEMENTS GUIDE

DASSAULT SYSTEMES

Contents

Trademarks and Legal Notices....6
Abaqus Elements Guide....7
What's New....8
Introduction....15
About the Element Library....16
Characterizing Elements....16
Combining Elements....18
Heat Transfer and Thermal-Stress Analysis....18
Information Available for Element Libraries....19
Choosing the Element's Dimensionality....20
One-Dimensional (Link) Elements....20
Two-Dimensional Elements....20
Three-Dimensional Elements....23
Cylindrical Elements....23
Axisymmetric Elements....23
Axisymmetric Elements with Nonlinear, Asymmetric Deformation....26
Choosing the Appropriate Element for an Analysis Type....27
Stress/Displacement Elements....27
Pore Pressure Elements....29
Coupled Temperature-Displacement Elements....30
Coupled Thermal-Electrical-Structural Elements....31
Coupled Temperature–Pore Pressure Elements....31
Diffusive (Heat Transfer) Elements....32
Forced Convection Heat Transfer Elements....33
Fluid Pipe and Fluid Pipe Connector Elements....33
Thermal Fluid Pipe and Thermal Fluid Pipe Connector Elements....34
Coupled Thermal-Electrical Elements....35
Coupled Thermal-Electrochemical Elements....35
Coupled Thermal-Electrochemical-Structural Elements....36
Coupled Thermal-Electrochemical-Structural–Pore Pressure Elements....37
Piezoelectric Elements....37
Electromagnetic Elements....38
Acoustic Elements....39
Poroelastic Acoustic Elements....39
Using the Same Mesh with Different Analysis or Element Types....40
Section Controls....41
Using Section Controls....42
Methods for Suppressing Hourglass Modes....43
Controlling Element Distortion for Crushable Materials in Abaqus/Explicit....48
Selecting a Scale Factor for the Drill Stiffness in Abaqus/Explicit....49
Drill Constraint in Small Strain Shell Elements S3RS and S4RS in Abaqus/Explicit....49
Ramping of Initial Stresses in Abaqus/Explicit....50
Defining the Kinematic Formulation for Hexahedron Solid Elements....50

Choosing the Order of Accuracy in Solid and Shell Element Formulations....52
Selecting Scale Factors for Bulk Viscosity in Abaqus/Explicit....52
Controlling the Activation of the "Improved" Element Time Estimation Method in Abaqus/Explicit....53
Controlling Element Deletion and Maximum Degradation for Materials with Damage Evolution....53
Controlling Shell Element Deletion Based on Integration Point Status in Abaqus/Explicit....54
Controlling Distortion-Based Element Deletion in Abaqus/Explicit....55
Using Linear Kinematic Conversion in Abaqus/Explicit....55
Using Section Controls for Slurry Transport in Abaqus/Standard....56
Using Viscous Damping with Connector Elements in Abaqus/Standard....57
Using Section Controls in an Import Analysis....57
Using Section Controls for Flexion-Torsion Type Connector....57
Using Section Controls to Define the Particle Tracking Box for DEM and SPH Particles....57
Using Section Controls for Smoothed Particle Hydrodynamics (SPH)....58
Using Section Controls for Bending Stabilization....62

Continuum Elements....64

General-Purpose Continuum Elements....65
Solid (Continuum) Elements....66
One-Dimensional Solid (Link) Element Library....86
Two-Dimensional Solid Element Library....92
Three-Dimensional Solid Element Library....114
Cylindrical Solid Element Library....144
Axisymmetric Solid Element Library....151
Axisymmetric Solid Elements with Nonlinear, Asymmetric Deformation....171
Infinite Elements....177
Infinite Elements....178
Infinite Element Library....186
Warping Elements....195
Warping Elements....196
Warping Element Library....198

Structural Elements....201

Membrane Elements....202
Membrane Elements....203
General Membrane Element Library....210
Cylindrical Membrane Element Library....218
Axisymmetric Membrane Element Library....223
Truss Elements....229
Truss Elements....230
Truss Element Library....233
Beam Elements....241
About Beam Modeling....242
Choosing a Beam Cross-Section....244
Choosing a Beam Element....247
Beam Element Cross-Section Orientation....256
Beam Section Behavior....261
Using a Beam Section Integrated during the Analysis to Define the Section Behavior....268
Using a General Beam Section to Define the Section Behavior....272
Beam Element Library....285

Beam Cross-Section Library....295
Frame Elements....313
Frame Elements....314
Frame Section Behavior....318
Frame Element Library....329
Elbow Elements....334
Pipes and Pipebends with Deforming Cross-Sections: Elbow Elements....335
Elbow Element Library....342
Shell Elements....348
About Shell Elements....349
Choosing a Shell Element....355
Defining the Initial Geometry of Conventional Shell Elements....365
Shell Section Behavior....369
Using a Shell Section Integrated during the Analysis to Define the Section Behavior....374
Using a General Shell Section to Define the Section Behavior....388
Three-Dimensional Conventional Shell Element Library....403
Continuum Shell Element Library....421
Axisymmetric Shell Element Library....433
Axisymmetric Shell Elements with Nonlinear, Asymmetric Deformation....443
Shear Panel Elements....448
Shear Panel Elements....449
Shear Panel Element Library....452
Inertial, Rigid, and Capacitance Elements....457
Point Mass Elements....458
Point Masses....459
Mass Element Library....461
Rotary Inertia Elements....463
Rotary Inertia....464
Rotary Inertia Element Library....466
Rigid Elements....468
Rigid Elements....469
Rigid Element Library....473
Capacitance Elements....482
Point Capacitance....483
Capacitance Element Library....484
Connector Elements....485
Connector Elements....486
About Connectors....487
Connector Elements....491
Connector Actuation....498
Connector Element Library....502
Connection Types....512
Connector Element Behavior....598
Connector Behavior....599
Connector Elastic Behavior....612
Connector Damping Behavior....618
Connector Functions for Coupled Behavior....623

Connector Friction Behavior....637
Connector Plastic Behavior....651
Connector Damage Behavior....663
Connector Stops and Locks....679
Connector Failure Behavior....683
Connector Uniaxial Behavior....686
Particle Elements....696
Discrete Particle Elements....697
Discrete Particle Elements....698
Discrete Particle Element Library....700
Continuum Particle Elements....702
Continuum Particle Elements....703
Continuum Particle Element Library....704
Special-Purpose Elements....706
Acoustic Interface Elements....707
Acoustic Interface Elements....708
Acoustic Interface Element Library....711
Cohesive Elements....715
About Cohesive Elements....716
Choosing a Cohesive Element....722
Modeling with Cohesive Elements....723
Defining the Cohesive Element's Initial Geometry....734
Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach....741
Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description....744
Defining the Constitutive Response of Fluid within the Cohesive Element Gap....763
Defining the Constitutive Response of Fluid Transitioning from Darcy Flow to Poiseuille Flow....769
Defining Thermal Convection for a Cohesive Element....778
Defining a Thermal Interaction for a Cohesive Element....781
Defining Slurry Transport and Placement within the Cohesive Element Gap....785
Two-Dimensional Cohesive Element Library....794
Three-Dimensional Cohesive Element Library....802
Axisymmetric Cohesive Element Library....812
Dashpot Elements....820
Dashpots....821
Dashpot Element Library....827
Distributing Coupling Elements....829
Distributing Coupling Elements....830
Distributing Coupling Element Library....834
Drag Chain Elements....836
Drag Chains....837
Drag Chain Element Library....839
Elastic-Plastic Joints....841
Elastic-Plastic Joints....842
Elastic-Plastic Joint Element Library....854
Eulerian Elements....856
Eulerian Elements....857
Eulerian Element Library....859
Flexible Joint Elements....866

Flexible Joint Element....867
Flexible Joint Element Library....869
Fluid Pipe Connector Elements....871
Fluid Pipe Connector Elements....872
Thermal Fluid Pipe Connector Elements....876
Slurry Fluid Pipe Connector Elements....881
Fluid Pipe Connector Element Library....885
Fluid Pipe Elements....892
Fluid Pipe Elements....893
Thermal Fluid Pipe Elements....898
Slurry Fluid Pipe Elements....904
Fluid Pipe Element Library....910
Gasket Elements....917
About Gasket Elements....918
Choosing a Gasket Element....922
Including Gasket Elements in a Model....925
Defining the Gasket Element's Initial Geometry....931
Defining the Gasket Behavior Using a Material Model....937
Defining the Gasket Behavior Directly Using a Gasket Behavior Model....939
Defining a Thermal Interaction for a Gasket Element....953
Two-Dimensional Gasket Element Library....956
Three-Dimensional Gasket Element Library....960
Axisymmetric Gasket Element Library....969
Line Spring Elements....973
Line Spring Elements for Modeling Part-through Cracks in Shells....974
Line Spring Element Library....978
Pipe-Soil Elements....982
Pipe-Soil Interaction Elements....983
Pipe-Soil Interaction Element Library....991
Spring Elements....993
Springs....994
Spring Element Library....1000
Surface Elements....1002
Surface Elements....1003
General Surface Element Library....1010
Cylindrical Surface Element Library....1017
Axisymmetric Surface Element Library....1022
Tube Support Elements....1027
Tube Support Elements....1028
Tube Support Element Library....1033
User-Defined Elements....1035
User-Defined Elements....1036
User-Defined Element Library....1051
Element Indexes....1053
Abaqus/Standard Element Index....1054
Abaqus/Explicit Element Index....1085

Trademarks and Legal Notices

Trademarks

Abaqus, **3D**EXPERIENCE®, the 3DS logo, the Compass icon, IFWE, 3DEXCITE, 3DVIA, BIOVIA, CATIA, CENTRIC PLM, DELMIA, ENOVIA, GEOVIA, MEDIDATA, NETVIBES, OUTSCALE, SIMULIA and SOLIDWORKS are commercial trademarks or registered trademarks of Dassault Systèmes, a European company (Societas Europaea) incorporated under French law, and registered with the Versailles trade and companies registry under number 322 306 440, or its subsidiaries in the United States and/or other countries. All other trademarks are owned by their respective owners. Use of any Dassault Systèmes or its subsidiaries trademarks is subject to their express written approval.

Legal Notices

Abaqus and this documentation may be used or reproduced only in accordance with the terms of the software license agreement signed by the customer, or, absent such agreement, the then current software license agreement to which the documentation relates.

This documentation and the software described in this documentation are subject to change without prior notice.

Dassault Systèmes or its Affiliates shall not be responsible for the consequences of any errors or omissions that may appear in this documentation.

© Dassault Systèmes Americas Corp., 2026.

For a full list of the third-party software contained in this release, please go to the Legal Notices in the Abaqus 2026 HTML documentation, which can be obtained from a documentation installation, or in the SIMULIA Established Products 2026 Program Directory, which is available on *www.3ds.com*.

Abaqus Elements Guide

The Abaqus Elements Guide describes the extensive element library that Abaqus provides as a powerful set of tools for solving many different problems.

This guide is a part of the Abaqus® documentation collection, which describes all the capabilities of the Abaqus finite element analysis technology used in SIMULIA® applications.

Five aspects of an element characterize its behavior:

- Family
- Degrees of freedom (directly related to the element family)
- Number of nodes
- Formulation
- Integration

Each element in Abaqus has a unique name, such as T2D2, S4R, C3D8I, or C3D8R. The element name identifies each of the five aspects of an element. For details on defining elements, see *Element Definition*.

What's New

This page describes recent changes in Abaqus Elements.

2026 GA

Coupled Temperature Displacement Elements with Incompatible Modes

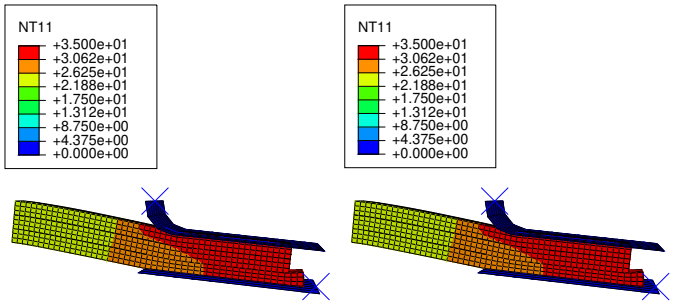
New incompatible elements with temperature degrees of freedom are now available in Abaqus/Standard (C3D8IT and C3D8IHT) and in Abaqus/Explicit (C3D8IT).

Benefits: The new elements provide improved bending behavior in coupled thermal-stress analysis of flexible structures.

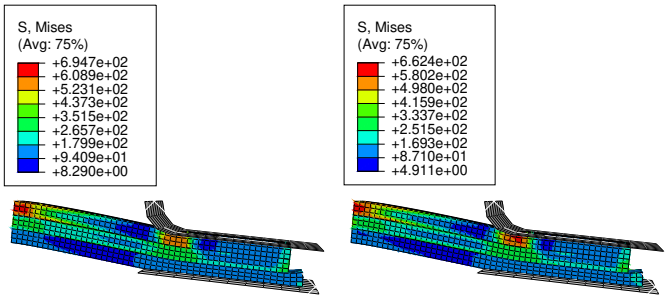
The incompatible mode element available in earlier releases, C3D8I, does not have temperature as a degree of freedom and cannot be used in heat transfer analyses. The C3D8T element has a temperature degree of freedom; however, this element is overly stiff when modeling bending deformation and is prone to shear locking. The new C3D8IT and C3D8IHT elements combine both temperature as a degree of freedom and incompatible modes, which makes them better options for modeling heat transfer in structures that undergo bending deformation.

The following example demonstrates the modeling of a beam-shaped structure in a coupled thermal-stress analysis. The beam structure contains three layers with different material properties, and contact gap conductance is defined between the adjacent layers. The bottom rigid surface holding the beam structure is fixed, and the top rigid surface is pushed down to introduce bending deformation of the beam structure during the analysis. A temperature boundary condition is also applied to the top surface of the beam structure once the top rigid surface is being pushed down.

For comparison purposes, both C3D8T and the new C3D8IT elements are used to model the beam structure. As shown below with C3D8T elements (left) and C3D8IT elements (right), the two thermal responses (node temperatures) are in close agreement.



However, the mechanical responses (stresses) are slightly different as shown below with C3D8T elements (left) and C3D8IT elements (right).



The C3D8T elements produce a stiffer response, which is expected since the beam structure undergoes bending deformation. The incompatible modes of the C3D8IT elements are activated during the analysis.

For more information, see *Three-Dimensional Solid Element Library*.

2025 FD01

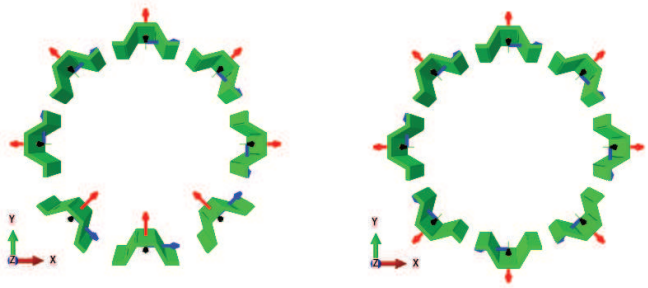
Fixed Nodal Normals for Beams in Space

You can now enforce the nodal normals for beams in space to be exactly as you specify them in the model definition.

Benefits: This new feature prevents nodal normals from getting reversed in certain special (rule-based) situations.

For beams in space you can define the nodal normal either by giving its direction cosines as part of each node definition or by giving them in a user-specified normal definition. By default, in certain situations, the nodal normal $\mathbf{n}_2$ that you specified might be automatically reversed by Abaqus (see *Defining Nodal Normals*). To prevent such default behavior, you can now enforce that the normal used by Abaqus is exactly as you defined it (and not its reverse).

In the example below, the user specified that all the beam normals point radially outward. However, as shown at left, by default three of the beam normals are automatically reversed by Abaqus during the preprocessing phase. These normals end up pointing radially inward, which leads to discontinuities in the beam normal in different regions of the model. You can use the new capability to prevent this from happening, as shown at right.



For more information, see **NORMAL* and *Beam Element Cross-Section Orientation*.

Surface Element Thickness

You can now define a constant section thickness as part of the surface section definition.

Benefits: Defining surface elements with a nonzero thickness enables you to better model contact in a similar way as shell and membrane elements.

User-defined surface elements have zero element thickness by default, but when used in contact, it is preferable to specify a thickness. In cases where surface elements are internally generated, those elements will have a nonzero thickness.

The specified thickness only affects contact calculations. Other element calculations related to rebar layers are unaffected.

For more information, see *Surface Elements* and **SURFACE SECTION*.

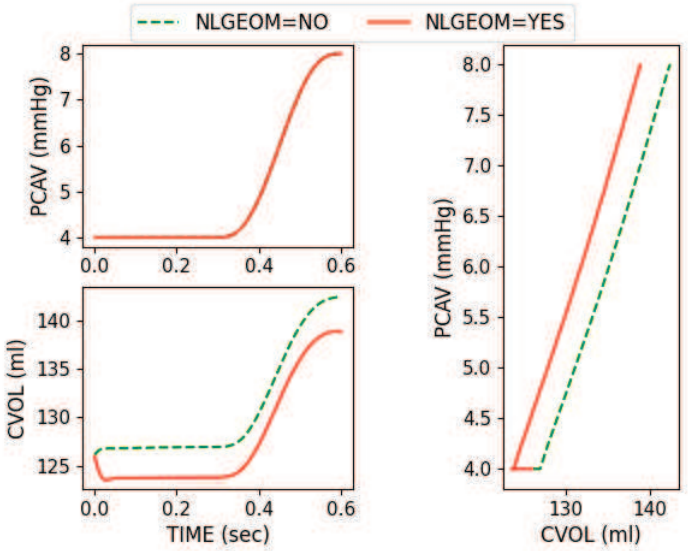
Geometrically Linear Analysis for C3D10 Elements

Geometrically linear analysis is now supported for C3D10 elements in Abaqus/Explicit.

Benefits: The ability to use C3D10 elements for the preload analysis of the Living Heart model provides more accurate initial stress states in the heart tissue.

The Living Heart model uses a preload analysis to compute stresses and corresponding stretches in the heart tissue corresponding to the initial loaded state. This preload analysis is performed using the geometrically linear option to calculate proper initial stress states in equilibrium with the applied loads in the reference configuration of the heart.

The following example demonstrates the effect of geometric linearity and geometric nonlinearity in the preload analysis on the pressure-volume response of a heart chamber. It consists of a preload analysis to compute the initial stress state of the heart tissue for the given applied cavity pressure, followed by a heartbeat analysis computing the response of the heart chamber as a function of the biological loads. Both geometric linearity and geometric nonlinearity are used to calculate the initial stress state in the preload analyses, which is then transferred to the subsequent heartbeat analyses. Pressure and volume versus time, as well as pressure versus volume results are compared between these two cases in the heartbeat analyses. As shown in the figure below, when geometric nonlinearity is used in the preload analysis, it shows a sharp drop in the heart chamber volume at the beginning of the heartbeat analysis; when geometric linearity is used in the preload analysis, the heart chamber volume results are more accurate and do not exhibit a sharp change.



For more information, see *Beginning a step in an Abaqus/Explicit analysis*.

PDF Guides Available

You can download PDF versions of the Abaqus guides from each guide overview.

Benefits: You can easily access the PDF versions of the guides.
In earlier releases, PDF versions of the guides were updated only for the GA (General Availability) release and could be downloaded from the Dassault Systèmes Knowledge Base. As of 2025 FD01, the PDF guides will be updated with each release of the HTML versions of the guides.

For more information, see *Abaqus Elements Guide*.

2024 FD03

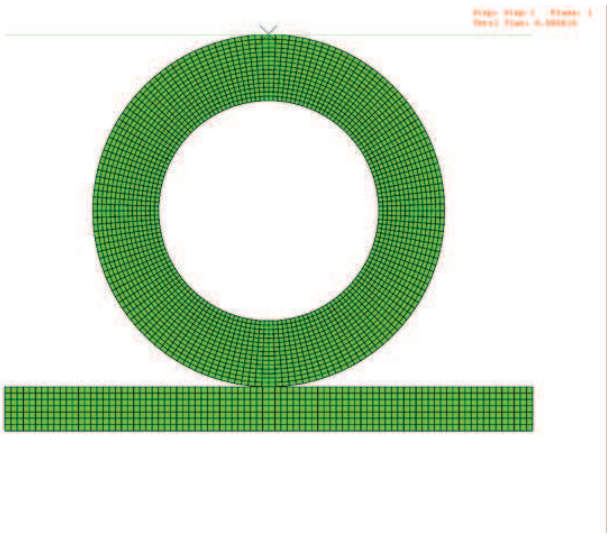
Bending Stabilization for Full-Integration Linear Elements

You can now control the activation and the level of bending stabilization for full-integration linear elements.

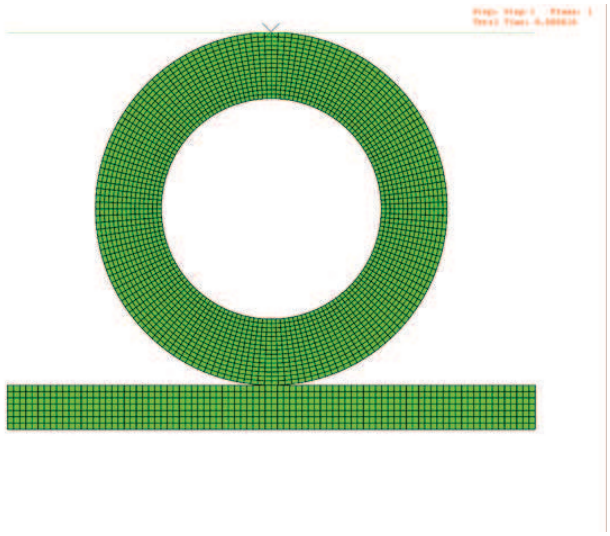
Benefits: Bending stabilization improves robustness for models using full-integration linear elements modeled with hyperelastic, hyperfoam, low-density foam, or user-defined materials and subjected to large bending deformations.

For full-integration linear elements modeled with hyperelastic, hyperfoam, low-density foam, or user-defined materials, spurious unstable patterns might develop in some situations due to excessively soft bending behavior. The formulation for this class of problems includes a bending stabilization term to improve the element performance. In earlier releases, the bending stabilization was active by default for built-in hyperelastic, hyperfoam, and low-density foam materials, but it was not active for user-defined materials. This could lead to different outcomes (convergence behavior and results) when you modeled the same material behavior using either a built-in material model or a user-defined material model. In addition, you had no control over the activation/deactivation of the stabilization algorithm or on the degree (magnitude) of stabilization.

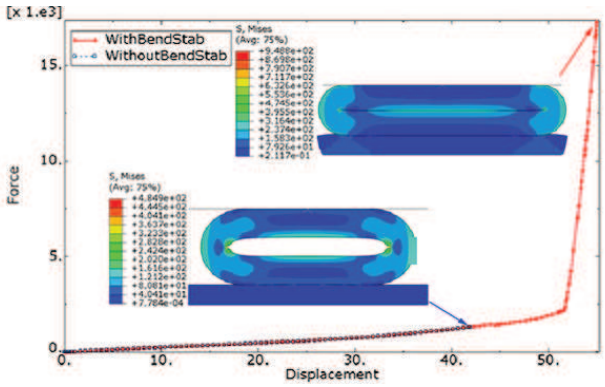
The example below shows a two-dimensional rubber ring compressed by a rigid plate against a base that is also made out of rubber. The image below shows the results from modeling a compressed O-ring that does not make use of the bending stabilization term. The analysis runs into convergence difficulties and fails to converge following the initiation of self-contact in the ring.



Conversely, the image below shows the results from a simulation that utilizes bending stabilization. In this situation, the analysis completes.



The image below shows the force-displacement behavior of the rigid plate’s reference node for the two models (with and without bending stabilization).



The two responses are in close agreement until convergence is lost in the model without stabilization, demonstrating that apart from improving convergence, the bending stabilization's impact on the overall model response is small. The bending stabilization term is very important for problems involving a relatively soft material subjected to large bending deformations.

For more information, see *Using Section Controls for Bending Stabilization*.

2024 FD01

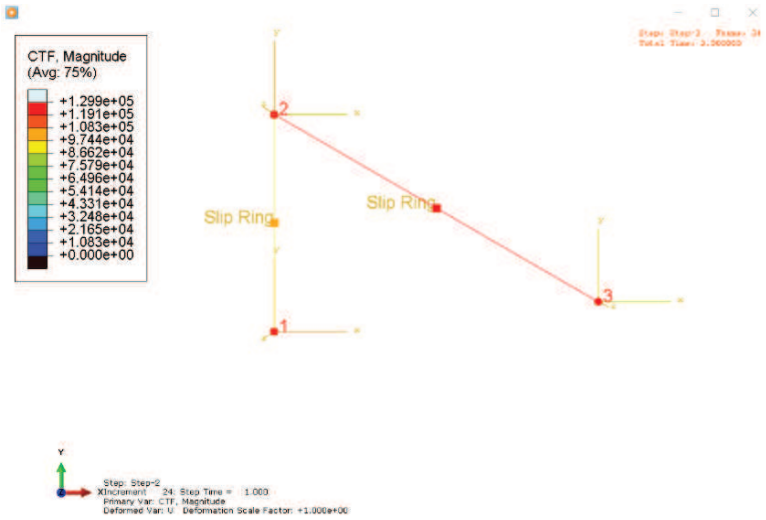
Varying Contact Angles for SLIPRING Connectors

You can now request that Abaqus/Standard compute the contact angle of a SLIPRING connector at the beginning of the analysis and update it through the analysis.

Benefits: If the slipring connector models a belt-pulley assembly, basing the calculations on the current value of the contact angle more accurately calculates the frictional force between the pulley and the belt.

In earlier releases, Abaqus/Standard assumed the SLIPRING contact angle to be either a constant (specified) value or zero (the default).

The animation below shows a belt-pulley system between nodes 1, 2, and 3.



The belt is modeled using two slipring connector elements and passes over a pulley at node 2. The frictional force between the belt and the pulley depends on the contact angle between them. In this example, the contact angle changes as nodes 1 and 3 are displaced from their original positions. Abaqus/Standard now accounts for the change in this contact angle for the friction calculations.

For more information, see *SLIPRING*.

Introduction

In this section:

- *About the Element Library*
- *Choosing the Element's Dimensionality*
- *Choosing the Appropriate Element for an Analysis Type*
- *Section Controls*

About the Element Library

Overview

Abaqus has an extensive element library to provide a powerful set of tools for solving many different problems.

References:

Characterizing Elements

Five aspects of an element characterize its behavior:

- Family
- Degrees of freedom (directly related to the element family)
- Number of nodes
- Formulation
- Integration

Each element in Abaqus has a unique name, such as T2D2, S4R, C3D8I, or C3D8R. The element name identifies each of the five aspects of an element. For details on defining elements, see *Element Definition*.

Family

Figure 1 shows the element families that are used most commonly in a stress analysis; in addition, continuum (fluid) elements are used in a fluid analysis. One of the major distinctions between different element families is the geometry type that each family assumes.

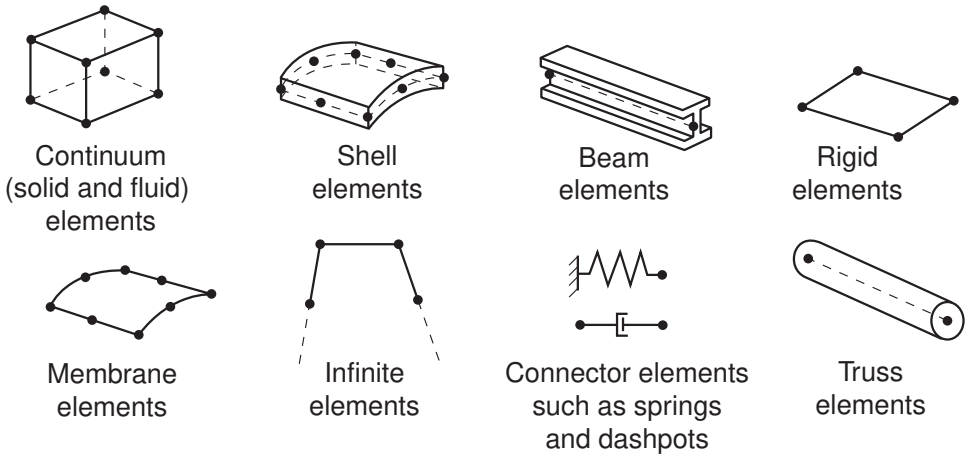


Figure 1: Commonly used element families.

The first letter or letters of an element's name indicate to which family the element belongs. For example, S4R is a shell element, CINPE4 is an infinite element, and C3D8I is a continuum element.

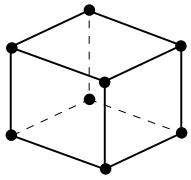
Degrees of Freedom

The degrees of freedom are the fundamental variables calculated during the analysis. For a stress/displacement simulation the degrees of freedom are the translations and, for shell, pipe, and beam elements, the rotations at each node. For a heat transfer simulation the degrees of freedom are the temperatures at each node; for a coupled thermal-stress analysis temperature degrees of freedom exist in addition to displacement degrees of freedom at each node. Heat transfer analyses and coupled thermal-stress analyses therefore require the use of different elements than does a stress analysis since the degrees of freedom are not the same. See *Conventions* for a summary of the degrees of freedom available in Abaqus for various element and analysis types.

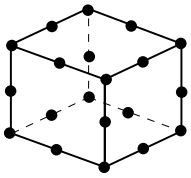
Number of Nodes and Order of Interpolation

Displacements or other degrees of freedom are calculated at the nodes of the element. At any other point in the element, the displacements are obtained by interpolating from the nodal displacements. Usually the interpolation order is determined by the number of nodes used in the element.

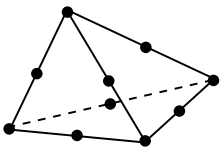
- Elements that have nodes only at their corners, such as the 8-node brick shown in *Figure 2*(a), use linear interpolation in each direction and are often called linear elements or first-order elements.
- In Abaqus/Standard elements with midside nodes, such as the 20-node brick shown in *Figure 2*(b), use quadratic interpolation and are often called quadratic elements or second-order elements.
- Modified triangular or tetrahedral elements with midside nodes, such as the 10-node tetrahedron shown in *Figure 2*(c), use a modified second-order interpolation and are often called modified or modified second-order elements.



(a) Linear element
(8-node brick, C3D8)



(b) Quadratic element
(20-node brick, C3D20)



(c) Modified second-order element
(10-node tetrahedron, C3D10M)

Figure 2: Linear brick, quadratic brick, and modified tetrahedral elements.

Typically, the number of nodes in an element is clearly identified in its name. The 8-node brick element is called C3D8, and the 4-node shell element is called S4R.

The beam element family uses a slightly different convention: the order of interpolation is identified in the name. Thus, a first-order, three-dimensional beam element is called B31, whereas a second-order, three-dimensional beam element is called B32. A similar convention is used for axisymmetric shell and membrane elements.

Formulation

An element's formulation refers to the mathematical theory used to define the element's behavior. In the Lagrangian, or material, description of behavior the element deforms with the material. In the alternative Eulerian, or spatial, description elements are fixed in space as the material flows through them. Eulerian methods are used commonly in fluid mechanics simulations. Abaqus/Standard uses Eulerian elements to model convective heat transfer. Abaqus/Explicit also offers multimaterial Eulerian elements for use in stress/displacement analyses. Adaptive meshing in Abaqus/Explicit combines the features of pure Lagrangian and Eulerian analyses and allows the motion of the element to be independent of the material (see *About ALE Adaptive Meshing*). All other

stress/displacement elements in Abaqus are based on the Lagrangian formulation. In Abaqus/Explicit the Eulerian elements can interact with Lagrangian elements through general contact (see *Eulerian Analysis*).

To accommodate different types of behavior, some element families in Abaqus include elements with several different formulations. For example, the conventional shell element family has three classes: one suitable for general-purpose shell analysis, another for thin shells, and yet another for thick shells. In addition, Abaqus also offers continuum shell elements, which have nodal connectivities like continuum elements but are formulated to model shell behavior with as few as one element through the shell thickness.

Some Abaqus/Standard element families have a standard formulation as well as some alternative formulations. Elements with alternative formulations are identified by an additional character at the end of the element name. For example, the continuum, beam, and truss element families include members with a hybrid formulation (to deal with incompressible or inextensible behavior); these elements are identified by the letter H at the end of the name (C3D8H or B31H).

Abaqus/Standard uses the lumped mass formulation for low-order elements; Abaqus/Explicit uses the lumped mass formulation for all elements. As a consequence, the second mass moments of inertia can deviate from the theoretical values, especially for coarse meshes.

In steady-state dynamic and frequency extraction procedures (see *About Dynamic Analysis Procedures*), Abaqus/Standard uses a special projected mass matrix algorithm for the S3, S3R, S4, S4R, SC6R, SC8R, and S4R5 shell elements. As a consequence, slight differences may be observed in models that contain these elements when comparing results from steady-state dynamic and frequency extraction procedures to those of an implicit dynamic analysis.

Integration

Abaqus uses numerical techniques to integrate various quantities over the volume of each element, thus allowing complete generality in material behavior. Using Gaussian quadrature for most elements, Abaqus evaluates the material response at each integration point in each element. Some continuum elements in Abaqus can use full or reduced integration, a choice that can have a significant effect on the accuracy of the element for a given problem.

Abaqus uses the letter R at the end of the element name to label reduced-integration elements. For example, CAX4R is the 4-node, reduced-integration, axisymmetric, solid element.

Shell, pipe, and beam element properties can be defined as general section behaviors; or each cross-section of the element can be integrated numerically, so that nonlinear response associated with nonlinear material behavior can be tracked accurately when needed. In addition, a composite layered section can be specified for shells and, in Abaqus/Standard, three-dimensional bricks, with different materials for each layer through the section.

Combining Elements

The element library is intended to provide a complete modeling capability for all geometries. Thus, any combination of elements can be used to make up the model; multi-point constraints (*General Multi-Point Constraints*) are sometimes helpful in applying the necessary kinematic relations to form the model (for example, to model part of a shell surface with solid elements and part with shell elements or to use a beam element as a shell stiffener).

Heat Transfer and Thermal-Stress Analysis

In cases where heat transfer analysis is to be followed by thermal-stress analysis, corresponding heat transfer and stress elements are provided in Abaqus/Standard. See *Sequentially Coupled Thermal-Stress Analysis* for additional details.

Information Available for Element Libraries

The complete element library in Abaqus is subdivided into a number of smaller libraries. Each library is presented as a separate section in this guide. In each of these sections, information regarding the following topics is provided where applicable:

- conventions;
- element types;
- degrees of freedom;
- nodal coordinates required;
- element property definition;
- element faces;
- element output;
- loading (general loading, distributed loads, foundations, distributed heat fluxes, film conditions, radiation types, distributed flows, distributed impedances, electrical fluxes, distributed electric current densities, and distributed concentration fluxes);
- nodes associated with the element;
- node ordering and face ordering on elements; and
- numbering of integration points for output.

For element libraries that are available in both Abaqus/Standard and Abaqus/Explicit, individual element or load types that are available only in Abaqus/Standard are designated with an (S); similarly, individual element or load types that are available only in Abaqus/Explicit are designated with an (E). Element or load types that are available in Abaqus/Aqua are designated with an (A).

Most of the element output variables available for an element are discussed. Additional variables may be available depending on the material model or the analysis procedure that is used. Some elements have solution variables that do not pertain to other elements of the same type; these variables are specified explicitly.

Choosing the Element's Dimensionality

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About the Element Library*
- *Part modeling space*
- *Assigning Abaqus element types*

Overview

The Abaqus element library contains the following for modeling a wide range of spatial dimensionality:

- one-dimensional elements;
- two-dimensional elements;
- three-dimensional elements;
- cylindrical elements;
- axisymmetric elements; and
- axisymmetric elements with nonlinear, asymmetric deformation.

One-Dimensional (Link) Elements

One-dimensional heat transfer, coupled thermal/electrical, and acoustic elements are available only in Abaqus/Standard. In addition, structural link (truss) elements are available in both Abaqus/Standard and Abaqus/Explicit. These elements can be used in two- or three-dimensional space to transmit loads or fluxes along the length of the element.

Two-Dimensional Elements

Abaqus provides several different types of two-dimensional elements. For structural applications these include plane stress elements and plane strain elements. Abaqus/Standard also provides generalized plane strain elements for structural applications.

Plane Stress Elements

Plane stress elements can be used when the thickness of a body or domain is small relative to its lateral (in-plane) dimensions. The stresses are functions of planar coordinates alone, and the out-of-plane normal and shear stresses are equal to zero.

Plane stress elements must be defined in the X–Y plane, and all loading and deformation are also restricted to this plane. This modeling method generally applies to thin, flat bodies. For anisotropic materials the Z-axis must be a principal material direction.

In Abaqus/Standard the output for the volume of a plane stress solid element is always the initial volume, irrespective of the level of deformation. However, the volume that is utilized for the finite element calculations depends on the material response. The formulation enforces kinematic incompressibility of the element when it

is used with a linear elastic material model (or nonlinear material models whose elastic behavior is defined as linear elastic) and the initial volume is used for all the calculations. This approach is consistent with the intended use of linear elastic materials for applications involving small elastic strains (and small volume changes) only. However, when plane stress solid elements are used with a compressible hyperelastic material model (or other material models whose elastic behavior is defined as compressible hyperelastic), the volume of the element evolves based on the compressibility of the material. The finite element calculations (that is, residual calculations) in this case utilize the current volume.

In Abaqus/Explicit the formulation of a plane stress solid element generally accounts for compressibility of the material response, and the current volume is used for all the calculations. The output of the element volume also corresponds to the current volume.

Plane Strain Elements

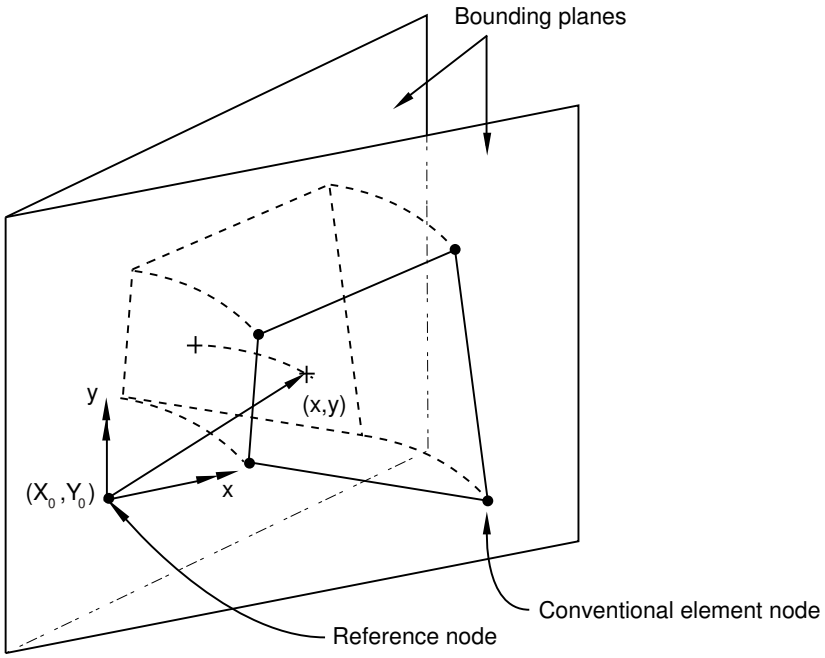
Plane strain elements can be used when it can be assumed that the strains in a loaded body or domain are functions of planar coordinates alone and the out-of-plane normal and shear strains are equal to zero.

Plane strain elements must be defined in the X–Y plane, and all loading and deformation are also restricted to this plane. This modeling method is generally used for bodies that are very thick relative to their lateral dimensions, such as shafts, concrete dams, or walls. Plane strain theory might also apply to a typical slice of an underground tunnel that lies along the Z-axis. For anisotropic materials the Z-axis must be a principal material direction.

Since plane strain theory assumes zero strain in the thickness direction, isotropic thermal expansion might cause large stresses in the thickness direction.

Generalized Plane Strain Elements

Generalized plane strain elements provide for the modeling of cases in Abaqus/Standard where the structure has constant curvature (and, hence, no gradients of solution variables) with respect to one material direction—the "axial" direction of the model. The formulation, thus, involves a model that lies between two planes that can move with respect to each other and, hence, cause strain in the axial direction of the model that varies linearly with respect to position in the planes, the variation being due to the change in curvature. In the initial configuration the bounding planes can be parallel or at an angle to each other, the latter case allowing the modeling of initial curvature of the model in the axial direction. The concept is illustrated in *Figure 1*. Generalized plane strain elements are typically used to model a section of a long structure that is free to expand axially or is subjected to axial loading.



Length of line through the thickness at (x,y) is

$$t_0 + \Delta u_z + \Delta\phi_x\,(y - Y_0) - \Delta\phi_y\,(x - X_0)$$

where quantities are defined in the text.

Figure 1: Generalized plane strain model.

Each generalized plane strain element has three, four, six, or eight conventional nodes, at each of which *x*- and *y*-coordinates, displacements, etc. are stored. These nodes determine the position and motion of the element in the two bounding planes. Each element also has a reference node, which is usually the same node for all the generalized plane strain elements in the model. The reference node of a generalized plane strain element should not be used as a conventional node in any element in the model. The reference node has three degrees of freedom 3, 4, and 5: (Δu_z, $\Delta\phi_x$, and $\Delta\phi_y$). The first degree of freedom (Δu_z) is the change in length of the axial material fiber connecting this node and its image in the other bounding plane. This displacement is positive as the planes move apart; therefore, there is a tensile strain in the axial fiber. The second and third degrees of freedom ($\Delta\phi_x$, $\Delta\phi_y$) are the components of the relative rotation of one bounding plane with respect to the other. The values stored are the two components of rotation about the *X*- and *Y*-axes in the bounding planes (that is, in the cross-section of the model). Positive rotation about the *X*-axis causes increasing axial strain with respect to the *y*-coordinate in the cross-section; positive rotation about the *Y*-axis causes decreasing axial strain with respect to the *x*-coordinate in the cross-section. The *x*- and *y*-coordinates of a generalized plane strain element reference node (X_0 and Y_0 discussed below) remain fixed throughout all steps of an analysis. From the degrees of freedom of the reference node, the length of the axial material fiber passing through the point with current coordinates (x, y) in a bounding plane is defined as

$$t = t_0 + \Delta u_z + (y - Y_0)\,\Delta\phi_x - (x - X_0)\,\Delta\phi_y,$$

where

t

is the current length of the fiber,

t_0

is the initial length of the fiber passing through the reference node (given as part of the element section definition),

Δu_z

is the displacement at the reference node (stored as degree of freedom 3 at the reference node),

$\Delta\phi_x$ and $\Delta\phi_y$

are the total values of the components of the angle between the bounding planes (the original values of $\Delta\phi_x$, $\Delta\phi_y$ are given as part of the element section definition—see *Defining the Elements Section Properties*: the changes in these values are the degrees of freedom 4 and 5 of the reference node), and

X_0 and Y_0

are the coordinates of the reference node in a bounding plane.

The strain in the axial direction is defined immediately from this axial fiber length. The strain components in the cross-section of the model are computed from the displacements of the regular nodes of the elements in the usual way. Since the solution is assumed to be independent of the axial position, there are no transverse shear strains.

Three-Dimensional Elements

Three-dimensional elements are defined in the global X, Y, Z space. These elements are used when the geometry and/or the applied loading are too complex for any other element type with fewer spatial dimensions.

Cylindrical Elements

Cylindrical elements are three-dimensional elements defined in the global X, Y, Z space. These elements are used to model bodies with circular or axisymmetric geometry subjected to general, nonaxisymmetric loading. Cylindrical elements are available only in Abaqus/Standard.

Cylindrical elements are useful in situations where the expected solution over a relatively large angle is nearly axisymmetric. In this case a very coarse mesh of cylindrical elements is often sufficient. Footprint and steady-state rolling analyses of tires are good examples of where cylindrical elements have distinct advantages over conventional continuum elements (see *Steady-state rolling analysis of a tire*). If, however, the expected solution has significant nonaxisymmetric components, a finer mesh of cylindrical elements will be needed and it might be more economical to use conventional continuum elements.

Axisymmetric Elements

Axisymmetric elements provide for the modeling of bodies of revolution under axially symmetric loading conditions. A body of revolution is generated by revolving a plane cross-section about an axis (the symmetry axis) and is readily described in cylindrical polar coordinates r, z, and θ. *Figure 2* shows a typical reference

cross-section at $\theta = 0$. The radial and axial coordinates of a point on this cross-section are denoted by r and z, respectively. At $\theta = 0$, the radial and axial coordinates coincide with the global Cartesian X- and Y-coordinates.

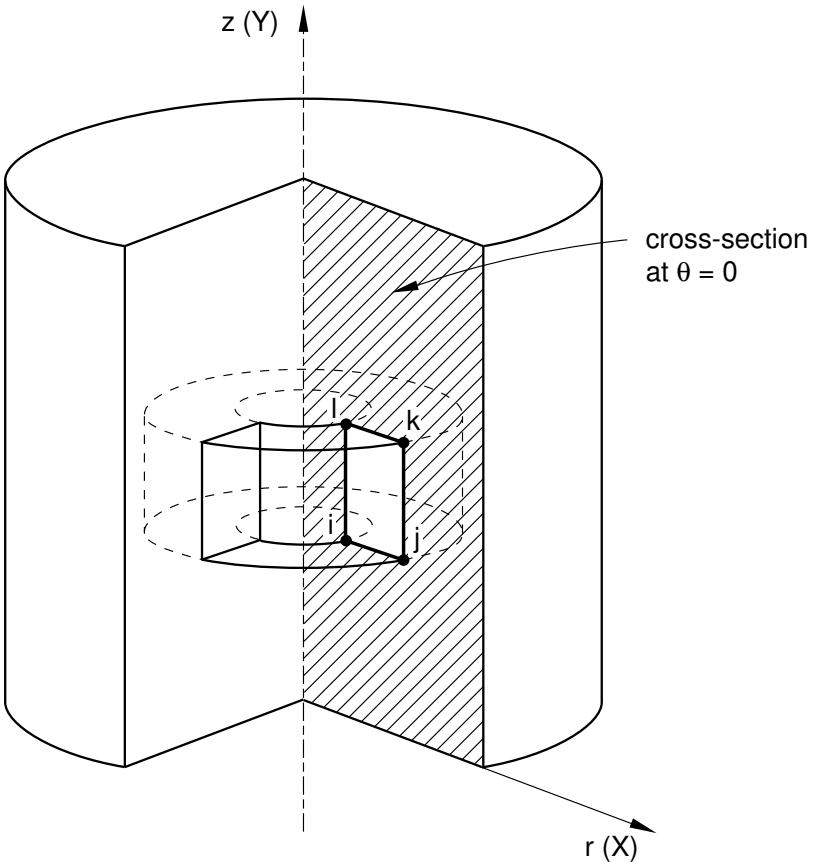


Figure 2: Reference cross-section and element in an axisymmetric solid.

Abaqus does not apply boundary conditions automatically to nodes that are located on the symmetry axis in axisymmetric models. If required, you should apply them directly. Radial boundary conditions at nodes located on the z-axis are appropriate for most problems because without them nodes might displace across the symmetry axis, violating the principle of compatibility. However, there are some analyses, such as penetration calculations, where nodes along the symmetry axis should be free to move; boundary conditions should be omitted in these cases.

If the loading and material properties are independent of θ, the solution in any r–z plane completely defines the solution in the body. Consequently, axisymmetric elements can be used to analyze the problem by discretizing the reference cross-section at $\theta = 0$. *Figure 2* shows an element of an axisymmetric body. The nodes i, j, k, and l are actually nodal "circles," and the volume of material associated with the element is that of a body of revolution, as shown in the figure. The value of a prescribed nodal load or reaction force is the total value on the ring; that is, the value integrated around the circumference.

Regular Axisymmetric Elements

Regular axisymmetric elements for structural applications allow for only radial and axial loading and have isotropic or orthotropic material properties, with θ being a principal direction. Any radial displacement in such an element will induce a strain in the circumferential direction ("hoop" strain); and since the displacement must also be purely axisymmetric, there are only four possible nonzero components of strain (ε_{rr}, ε_{zz}, $\varepsilon_{\theta\theta}$, and γ_{rz}).

Generalized Axisymmetric Stress/Displacement Elements with Twist

Axisymmetric solid elements with twist are available only in Abaqus/Standard for the analysis of structures that are axially symmetric but can twist about their symmetry axis. This element family is similar to the axisymmetric elements discussed above, except that it allows for a circumferential loading component (which is independent of θ) and for general material anisotropy. Under these conditions, there might be displacements in the θ-direction that vary with r and z but not with θ. The problem remains axisymmetric because the solution does not vary as a function of θ so that the deformation of any r–z plane characterizes the deformation in the entire body. Initially the elements define an axisymmetric reference geometry with respect to the r–z plane at $\theta = 0$, where the r-direction corresponds to the global X-direction and the z-direction corresponds to the global Y-direction. *Figure 3* shows an axisymmetric model consisting of two elements. The figure also shows the local cylindrical coordinate system at node 100.

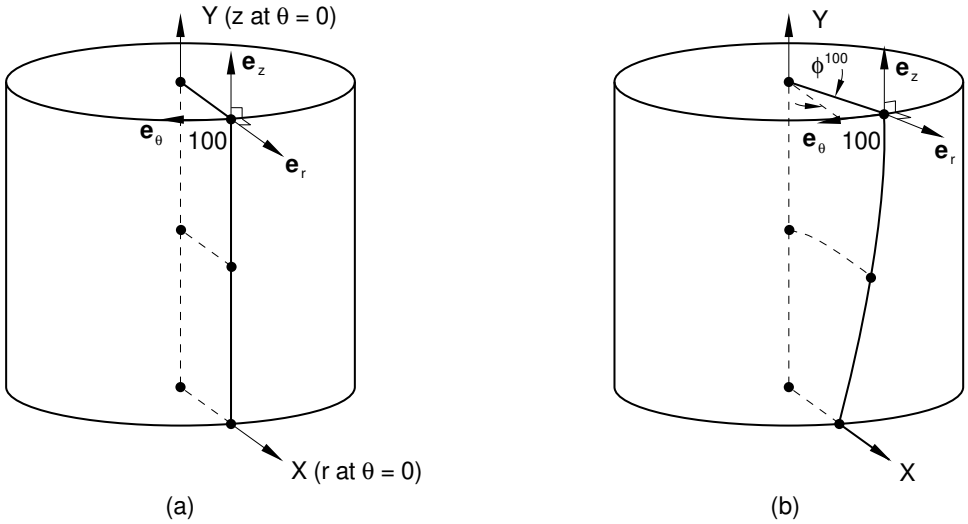


Figure 3: Reference and deformed cross-section in an axisymmetric solid with twist.

The motion at a node of an axisymmetric element with twist is described by the radial displacement u_r, the axial displacement u_z, and the twist ϕ (in radians) about the z-axis, each of which is constant in the circumferential direction, so that the deformed geometry remains axisymmetric. *Figure 3*(b) shows the deformed geometry of the reference model shown in *Figure 3*(a) and the local cylindrical coordinate system at the displaced location of node 100, for a twist ϕ^{100}.

The formulation of these elements is discussed in *Axisymmetric elements*.

Generalized axisymmetric elements with twist cannot be used in contour integral calculations and in dynamic analysis. Elastic foundations are applied only to degrees of freedom u_r and u_z.

These elements should not be mixed with three-dimensional elements.

Axisymmetric elements with twist and the nodes of these elements should be used with caution within rigid bodies. If the rigid body undergoes large rotations, incorrect results might be obtained. It is recommended that rigid constraints on axisymmetric elements with twist be modeled with kinematic coupling (see *Kinematic Coupling Constraints*).

Stabilization should not be used with these elements if the deformation is dominated by twist, since stabilization is applied only to the in-plane deformation.

Axisymmetric Elements with Nonlinear, Asymmetric Deformation

These elements are intended for the linear or nonlinear analysis of structures that are initially axisymmetric but undergo nonlinear, nonaxisymmetric deformation. They are available only in Abaqus/Standard.

The elements use standard isoparametric interpolation in the *r*–*z* plane, combined with Fourier interpolation with respect to θ. The deformation is assumed to be symmetric with respect to the *r*–*z* plane at $\theta = 0, \pi$.

Up to four Fourier modes are allowed. For more general cases, full three-dimensional modeling or cylindrical element modeling is probably more economical because of the complete coupling between all deformation modes.

These elements use a set of nodes in each of several *r*–*z* planes: the number of such planes depends on the order *N* of Fourier interpolation used with respect to θ, as follows:

Number of Fourier modes *N*	Number of nodal planes	Nodal plane locations with respect to θ
1	2	$0, \pi$
2	3	$0, \pi/2, \pi$
3	4	$0, \pi/3, 2\pi/3, \pi$
4	5	$0, \pi/4, \pi/2, 3\pi/4, \pi$

Each element type is defined by a name such as CAXA8R*N* (continuum elements) or SAXA1*N* (shell elements). The number *N* should be given as the number of Fourier modes to be used with the element (*N*=1, 2, 3, or 4). For example, element type CAXA8R2 is a quadrilateral in the *r*–*z* plane with biquadratic interpolation in this plane and two Fourier modes for interpolation with respect to θ. The nodal planes associated with various Fourier modes are illustrated in *Figure 4*.

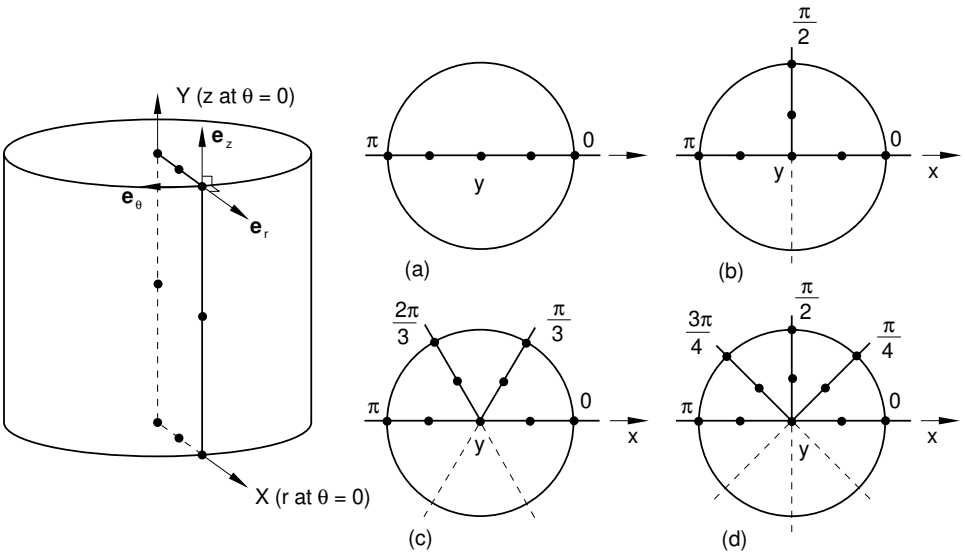


Figure 4: Nodal planes of a second-order axisymmetric element with nonlinear, asymmetric deformation and (a) 1, (b) 2, (c) 3, or (d) 4 Fourier modes.

Choosing the Appropriate Element for an Analysis Type

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About the Element Library*
- *Element type assignment*

Overview

Abaqus has an extensive element library.

The Abaqus element library contains the following:

- stress/displacement elements, including contact elements, connector elements such as springs, and special-purpose elements such as Eulerian elements and surface elements;
- pore pressure elements;
- coupled temperature-displacement elements;
- coupled thermal-electrical-structural elements;
- coupled temperature–pore pressure displacement elements;
- heat transfer or mass diffusion elements;
- forced convection heat transfer elements;
- fluid pipe and fluid pipe connector elements;
- thermal fluid pipe and thermal fluid pipe connector elements;
- coupled thermal-electrical elements;
- coupled thermal-electrochemical elements;
- coupled thermal-electrochemical-structural elements;
- coupled thermal-electrochemical-structural–pore pressure elements;
- piezoelectric elements;
- electromagnetic elements;
- acoustic elements;
- poroelastic acoustic elements; and
- user-defined elements.

Within Abaqus/Standard or Abaqus/Explicit, a model can contain elements that are not appropriate for the particular analysis type chosen; such elements will be ignored. However, an Abaqus/Standard model cannot contain elements that are not available in Abaqus/Standard; likewise, an Abaqus/Explicit model cannot contain elements that are not available in Abaqus/Explicit.

Stress/Displacement Elements

Stress/displacement elements are used in the modeling of linear or complex nonlinear mechanical analyses that possibly involve contact, plasticity, and/or large deformations. Stress/displacement elements can also be used for thermal-stress analysis, where the temperature history can be obtained from a heat transfer analysis carried out with diffusive elements.

Analysis Types

Stress/displacement elements can be used in the following analysis types:

- static and quasi-static analysis (*About Static Stress Analysis Procedures*);
- implicit transient dynamic, explicit transient dynamic, modal dynamic, and steady-state dynamic analysis (*About Dynamic Analysis Procedures*);
- *Coupled Acoustic-Structural Analysis*; and
- *About Fracture Mechanics*.

Active Degrees of Freedom

Stress/displacement elements have only displacement degrees of freedom. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Choosing a Stress/Displacement Element

Stress/displacement elements are available in several different element families.

Continuum Elements

- *Solid (Continuum) Elements*; and
- *Infinite Elements*.

Structural Elements

- *Membrane Elements*;
- *Truss Elements*;
- *About Beam Modeling*;
- *Frame Elements*;
- *Pipes and Pipebends with Deforming Cross-Sections: Elbow Elements*;
- *About Shell Elements*; and
- *Shear Panel Elements*.

Rigid Elements

- *Point Masses*;
- *Rotary Inertia*; and
- *Rigid Elements*.

Connector Elements

- *Connector Elements*;
- *Springs*;
- *Dashpots*;
- *Flexible Joint Element*;

- *Tube Support Elements*; and
- *Drag Chains*.

Special-Purpose Elements

- *About Cohesive Elements*;
- *About Gasket Elements*;
- *Surface Elements*;
- *Line Spring Elements for Modeling Part-through Cracks in Shells*;
- *Elastic-Plastic Joints*;
- *Eulerian Elements*;
- *Fluid Pipe Connector Elements*;
- *Fluid Pipe Elements*;
- *Thermal Fluid Pipe Connector Elements*; and
- *Thermal Fluid Pipe Elements*.

Contact Elements

- *Gap Contact Elements*;
- *Tube-to-Tube Contact Elements*;
- *Slide Line Contact Elements*; and
- *Rigid Surface Contact Elements*.

Pore Pressure Elements

Pore pressure elements are provided in Abaqus/Standard for modeling fully or partially saturated fluid flow through a deforming porous medium. The names of all pore pressure elements include the letter P (pore pressure). These elements cannot be used with hydrostatic fluid elements.

Analysis Types

Pore pressure elements can be used in the following analysis types:

- soils analysis (*Coupled Pore Fluid Diffusion and Stress Analysis*); and
- geostatic analysis (*Geostatic Stress State*).

Active Degrees of Freedom

Pore pressure elements have both displacement and pore pressure degrees of freedom. In second-order elements the pore pressure degrees of freedom are active only at the corner nodes. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Interpolation

These elements use either linear- or second-order (quadratic) interpolation for the geometry and displacements in two or three directions. The pore pressure is interpolated linearly from the corner nodes. Curved element edges should be avoided; exact linear spatial pore pressure variations cannot be obtained with curved edges.

For output purposes, the pore pressure at the midside nodes of second-order elements is determined by linear interpolation from the corner nodes.

Choosing a Pore Pressure Element

Pore pressure elements are available only in the following element family:

- *Solid (Continuum) Elements*.

Coupled Temperature-Displacement Elements

Coupled temperature-displacement elements are used in problems for which the stress analysis depends on the temperature solution and the thermal analysis depends on the displacement solution. An example is the heating of a deforming body whose properties are temperature dependent by plastic dissipation or friction. The names of all coupled temperature-displacement elements include the letter T.

Analysis Types

Coupled temperature-displacement elements are for use in fully coupled temperature-displacement analysis (*Fully Coupled Thermal-Stress Analysis*).

Active Degrees of Freedom

Coupled temperature-displacement elements have both displacement and temperature degrees of freedom. In second-order elements the temperature degrees of freedom are active at the corner nodes (except for C3D10T elements in Abaqus/Explicit, for which the temperature degrees of freedom are active at every node). In modified triangle and tetrahedron elements the temperature degrees of freedom are active at every node. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Interpolation

Coupled temperature-displacement elements use either linear or parabolic interpolation for the geometry and displacements. The temperature is always interpolated linearly except for C3D10T elements in Abaqus/Explicit, for which parabolic interpolation is used. In second-order elements (except for C3D10T elements in Abaqus/Explicit), curved edges should be avoided; exact linear spatial temperature variations for these elements cannot be obtained with curved edges.

In Abaqus/Standard, for output purposes the temperature at the midside nodes of second-order elements is determined by linear interpolation from the corner nodes.

Choosing a Coupled Temperature-Displacement Element

Coupled temperature-displacement elements are available in the following element families:

- *Solid (Continuum) Elements*;
- *Truss Elements*;
- *About Shell Elements*;
- *Gap Contact Elements*; and
- *Slide Line Contact Elements*.

Coupled Thermal-Electrical-Structural Elements

Coupled thermal-electrical-structural elements are used when a solution for the displacement, electrical potential, and temperature degrees of freedom must be obtained simultaneously. In these types of problems, coupling between the temperature and displacement degrees of freedom arises from temperature-dependent material properties, thermal expansion, and internal heat generation, which is a function of inelastic deformation of the material. The coupling between the temperature and electrical degrees of freedom arises from temperature-dependent electrical conductivity and internal heat generation (Joule heating), which is a function of the electrical current density. The names of the coupled thermal-electrical-structural elements begin with the letter Q.

Analysis Types

Coupled thermal-electrical-structural elements are for use in a fully coupled thermal-electrical-structural analysis (*Fully Coupled Thermal-Electrical-Structural Analysis*).

Active Degrees of Freedom

Coupled thermal-electrical-structural elements have displacement, electrical potential, and temperature degrees of freedom. In second-order elements the electrical potential and temperature degrees of freedom are active at the corner nodes. In modified tetrahedron elements the electrical potential and temperature degrees of freedom are active at every node. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Interpolation

Coupled thermal-electrical-structural elements use either linear or parabolic interpolation for the geometry and displacements. The electrical potential and temperature are always interpolated linearly. In second-order elements curved edges should be avoided; exact linear spatial electrical potential and temperature variations for these elements cannot be obtained with curved edges.

For output purposes, the electrical potential and temperature at the midside nodes of second-order elements are determined by linear interpolation from the corner nodes.

Choosing a Coupled Thermal-Electrical-Structural Element

Coupled thermal-electrical-structural elements are available only in the following element family:

- *Solid (Continuum) Elements*.

Coupled Temperature–Pore Pressure Elements

Coupled temperature–pore pressure elements are used in Abaqus/Standard for modeling fully or partially saturated fluid flow through a deforming porous medium in which the stress, fluid pore pressure, and temperature fields are fully coupled to one another. The names of all coupled temperature–pore pressure elements include the letters T and P. These elements cannot be used with hydrostatic fluid elements.

Analysis Types

Coupled temperature–pore pressure elements are for use in fully coupled temperature–pore pressure analysis (*Coupled Pore Fluid Diffusion and Stress Analysis*).

Active Degrees of Freedom

Coupled temperature–pore pressure elements have displacement, pore pressure, and temperature degrees of freedom. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Interpolation

These elements use either linear- or second-order (quadratic) interpolation for the geometry and displacements. The temperature and pore pressure are always interpolated linearly.

Choosing a Coupled Temperature–Pore Pressure Element

Coupled temperature–pore pressure elements are available in the following element family:

- *Solid (Continuum) Elements*.

Diffusive (Heat Transfer) Elements

Diffusive elements are provided in Abaqus/Standard for use in heat transfer analysis (*Uncoupled Heat Transfer Analysis*), where they allow for heat storage (specific heat and latent heat effects) and heat conduction. They provide temperature output that can be used directly as input to the equivalent stress elements. The names of all diffusive heat transfer elements begin with the letter D.

Analysis Types

The diffusive elements can be used in mass diffusion analysis (*Mass Diffusion Analysis*) as well as in heat transfer analysis.

Active Degrees of Freedom

When used for heat transfer analysis, the diffusive elements have only temperature degrees of freedom. When they are used in a mass diffusion analysis, they have normalized concentration, instead of temperature, degrees of freedom. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Interpolation

The diffusive elements use either first-order (linear) interpolation or second-order (quadratic) interpolation in one, two, or three dimensions.

Choosing a Diffusive Element

Diffusive elements are available in the following element families:

- *Solid (Continuum) Elements*;
- *About Shell Elements* (these elements cannot be used in a mass diffusion analysis); and
- *Gap Contact Elements*.

Forced Convection Heat Transfer Elements

Forced convection heat transfer elements are provided in Abaqus/Standard to allow for heat storage (specific heat) and heat conduction, as well as the convection of heat by a fluid flowing through the mesh (forced convection). All forced convection heat transfer elements provide temperature output, which can be used directly as input to the equivalent stress elements. The names of all forced convection heat transfer elements begin with the letters DCC.

Analysis Types

The forced convection heat transfer elements can be used in heat transfer analyses (*Uncoupled Heat Transfer Analysis*), including cavity radiation modeling (*Cavity Radiation in Abaqus/Standard*). The forced convection heat transfer elements can be used together with the diffusive elements.

Active Degrees of Freedom

The forced convection heat transfer elements have temperature degrees of freedom. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Interpolation

The forced convection heat transfer elements use only first-order (linear) interpolation in one, two, or three dimensions.

Choosing a Forced Convection Heat Transfer Element

Forced convection heat transfer elements are available only in the following element family:

- *Solid (Continuum) Elements*.

Fluid Pipe and Fluid Pipe Connector Elements

Fluid pipe elements suitable for modeling incompressible pipe flow and fluid pipe connector elements suitable for modeling the junction between two pipes are available in Abaqus/Standard. These elements have only pore pressure degree of freedom. The names of all fluid pipe elements begin with the letters FP. The names of all fluid pipe connector elements begin with the letters FPC.

Analysis Types

The fluid pipe and fluid pipe connector elements can be used in the following analyses:

- soils analysis (*Coupled Pore Fluid Diffusion and Stress Analysis*); and
- geostatic analysis (*Geostatic Stress State*).

Active Degrees of Freedom

The fluid pipe and fluid pipe connector elements provide primarily pore pressure degree of freedom. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Choosing a Fluid Pipe Element

The fluid pipe elements are available only in the following element family:

- *Fluid Pipe Elements*.

Choosing a Fluid Pipe Connector Element

The fluid pipe connector elements are available only in the following element family:

- *Fluid Pipe Connector Elements*.

Thermal Fluid Pipe and Thermal Fluid Pipe Connector Elements

Thermal fluid pipe elements suitable for modeling incompressible pipe flow and temperature fields and thermal fluid pipe connector elements suitable for modeling the junction between two or more thermal fluid pipes are available in Abaqus/Standard. These elements have pore pressure degree of freedom to model the fluid flowing in the pipe. In addition, temperature degrees of freedom are present at all nodes to model the evolution of temperatures in the fluid and pipe walls. The names of all thermal fluid pipe elements begin with the letters FP. The names of all thermal fluid pipe connector elements begin with the letters FPC.

Analysis Types

The thermal fluid pipe and thermal fluid pipe connector elements can be used in the following analyses:

- soils analysis (*Coupled Pore Fluid Diffusion and Stress Analysis*); and
- geostatic analysis (*Geostatic Stress State*).

Active Degrees of Freedom

The thermal fluid pipe and thermal fluid pipe connector elements provide primarily pore pressure and temperature degrees of freedom. See *Conventions* for a discussion of the degrees of freedom in Abaqus. The pore pressure degree of freedom is present only on the nodes modeling the fluid behavior, while the temperature degree of freedom is present at all nodes.

Choosing a Thermal Fluid Pipe Element

The thermal fluid pipe elements are available only in the following element family:

- *Thermal Fluid Pipe Elements*.

Choosing a Thermal Fluid Pipe Connector Element

The thermal fluid pipe connector elements are available only in the following element family:

- *Thermal Fluid Pipe Connector Elements*.

Coupled Thermal-Electrical Elements

Coupled thermal-electrical elements are provided in Abaqus/Standard for use in modeling heating that arises when an electrical current flows through a conductor (Joule heating).

Analysis Types

The Joule heating effect requires full coupling of the thermal and electrical problems (see *Coupled Thermal-Electrical Analysis*). The coupling arises from two sources: temperature-dependent electrical conductivity and the heat generated in the thermal problem by electric conduction.

These elements can also be used to perform uncoupled electric conduction analysis in all or part of the model. In such analysis only the electric potential degree of freedom is activated, and all heat transfer effects are ignored. This capability is available by omitting the thermal conductivity from the material definition.

The coupled thermal-electrical elements can also be used in heat transfer analysis (*Uncoupled Heat Transfer Analysis*), in which case all electric conduction effects are ignored. This feature is quite useful if a coupled thermal-electrical analysis is followed by a pure heat conduction analysis (such as a welding simulation followed by cool down).

The elements cannot be used in any of the stress/displacement analysis procedures.

Active Degrees of Freedom

Coupled thermal-electrical elements have both temperature and electrical potential degrees of freedom. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Interpolation

Coupled thermal-electrical elements are provided with first- or second-order interpolation of the temperature and electrical potential.

Choosing a Coupled Thermal-Electrical Element

Coupled thermal-electrical elements are available only in the following element family:

- *Solid (Continuum) Elements*.

Coupled Thermal-Electrochemical Elements

Coupled thermal-electrochemical elements are provided in Abaqus/Standard for use in modeling battery electrochemistry.

Analysis Types

The modeling of battery electrochemistry requires full coupling of the thermal, electrical, and electrochemical problems (see *Coupled Thermal-Electrochemical Analysis* and *Modeling Solid Electrolytes and Solid-State Batteries*). The coupling arises from the flow of electrons and ions in the solid and electrolyte phases of the battery and the intercalation process at the solid-liquid interface.

You can use the coupled thermal-electrochemical elements only in the coupled thermal-electrochemical analysis procedure.

Active Degrees of Freedom

Coupled thermal-electrochemical elements have temperature, electrical potential in the solid and electrolyte phases, ion concentration, and (for solid electrolytes and solid-state batteries) species concentration degrees of freedom. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Interpolation

Coupled thermal-electrochemical elements are provided with first-order interpolation of the five fields; namely, temperature, electrical potential in the solid and electrolyte phases, ion concentration, and species concentration.

Choosing a Coupled Thermal-Electrochemical Element

Coupled thermal-electrochemical elements are available only in the following element family:

- *Solid (Continuum) Elements*.

Coupled Thermal-Electrochemical-Structural Elements

Coupled thermal-electrochemical-structural elements are provided in Abaqus/Standard for use in modeling battery electrochemistry.

Analysis Types

The modeling of battery electrochemistry allows for full coupling of the structural, thermal, electrical, and electrochemical problems (see *Fully Coupled Thermal-Electrochemical-Structural Analysis* and *Modeling Solid Electrolytes and Solid-State Batteries*). The coupling arises from the flow of electrons and ions in the solid and electrolyte phases of the battery and the intercalation process at the solid-liquid interface. The coupling to the mechanical degrees of freedom arises from thermal effects and particle swelling during the intercalation/deintercalation process.

You can use the coupled thermal-electrochemical-structural elements only in the coupled thermal-electrochemical-structural analysis procedure.

Active Degrees of Freedom

Coupled thermal-electrochemical-structural elements have displacement, temperature, electrical potential in the solid and electrolyte phases, ion concentration, and (for solid electrolytes and solid-state batteries) species concentration degrees of freedom. The temperature degree of freedom is inactive if thermal conductivity is omitted from the material definition. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Interpolation

Coupled thermal-electrochemical-structural elements are provided with first-order interpolation of the five fields; namely, displacements, temperature, electrical potential in the solid and electrolyte phases, ion concentration, and species concentration.

Choosing a Coupled Thermal-Electrochemical-Structural Element

Coupled thermal-electrochemical-structural elements are available only in the following element family:

- *Solid (Continuum) Elements*.

Coupled Thermal-Electrochemical-Structural–Pore Pressure Elements

Coupled thermal-electrochemical-structural–pore pressure elements are provided in Abaqus/Standard for use in modeling battery electrochemistry taking into account the effects of electrolyte fluid flow within the porous electrodes and separator.

Analysis Types

The modeling of battery electrochemistry allows for full coupling of the structural, pore pressure, thermal, electrical, and electrochemical problems (see *Fully Coupled Thermal-Electrochemical-Structural–Pore Pressure Analysis*). The electrochemical coupling arises from the interplay among the flow of electrons and ions in the solid and electrolyte phases of the battery, respectively, and the intercalation process at the solid-liquid interface. The thermal coupling arises from Joule and entropic heat generation in the different components of the battery. The mechanical coupling is the result of thermal expansion in the solid and electrolyte phases, as well as lithium particle expansion (contraction) during the intercalation (deintercalation) process. The pore pressure coupling is the result of the interplay between the deformation of the solid phases and the associated pore pressure–mediated electrolyte flow in various parts of the battery.

You can use the coupled thermal-electrochemical-structural–pore pressure elements only in the coupled thermal-electrochemical-structural–pore pressure analysis procedure.

Active Degrees of Freedom

Coupled thermal-electrochemical-structural–pore pressure elements have the following degrees of freedom: displacement, pore pressure, temperature, electrical potential in the solid phase, electrical potential in the electrolyte phase, and ion concentration. The temperature degree of freedom is inactive if thermal conductivity is omitted from the material definition. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Interpolation

Coupled thermal-electrochemical-structural–pore pressure elements are provided with first-order interpolation of the six fields; that is, displacement, pore pressure, temperature, electrical potential in the solid and electrolyte phases, and ion concentration.

Choosing a Coupled Thermal-Electrochemical-Structural–Pore Pressure Element

Coupled thermal-electrochemical-structural–pore pressure elements are available only in the following element family:

- *Solid (Continuum) Elements*.

Piezoelectric Elements

Piezoelectric elements are provided in Abaqus/Standard for problems in which a coupling between the stress and electrical potential (the piezoelectric effect) must be modeled.

Analysis Types

Piezoelectric elements are for use in piezoelectric analysis (*Piezoelectric Analysis*).

Active Degrees of Freedom

The piezoelectric elements have both displacement and electric potential degrees of freedom. See *Conventions* for a discussion of the degrees of freedom in Abaqus. The piezoelectric effect is discussed further in *Piezoelectric Analysis*.

Interpolation

Piezoelectric elements are available with first- or second-order interpolation of displacement and electrical potential.

Choosing a Piezoelectric Element

Piezoelectric elements are available in the following element families:

- *Solid (Continuum) Elements*; and
- *Truss Elements*.

Electromagnetic Elements

Electromagnetic elements are provided in Abaqus/Standard for problems that require the computation of the magnetic fields (such as a magnetostatic analysis) or for problems in which a coupling between electric and magnetic fields must be modeled (such as an eddy current analysis).

Analysis Types

Electromagnetic elements are for use in magnetostatic and eddy current analyses (*Magnetostatic Analysis* and *Eddy Current Analysis*).

Active Degrees of Freedom

Electromagnetic elements have magnetic vector potential as the degree of freedom. See *Conventions* for a discussion of the degrees of freedom in Abaqus. Magnetostatic analysis is discussed further in *Magnetostatic Analysis*, while the electromagnetic coupling that occurs in an eddy current analysis is discussed further in *Eddy Current Analysis*.

Interpolation

Electromagnetic elements are available with zero-order element edge–based interpolation of the magnetic vector potential.

Choosing an Electromagnetic Element

Electromagnetic elements are available in the following element family:

- *Solid (Continuum) Elements*.

Acoustic Elements

Acoustic elements are used for modeling an acoustic medium undergoing small pressure changes. The solution in the acoustic medium is defined by a single pressure variable. Impedance boundary conditions representing absorbing surfaces or radiation to an infinite exterior are available on the surfaces of these acoustic elements.

Acoustic infinite elements, which improve the accuracy of analyses involving exterior domains, and acoustic-structural interface elements, which couple an acoustic medium to a structural model, are also provided.

Analysis Types

Acoustic elements are for use in acoustic and coupled acoustic-structural analysis (*Coupled Acoustic-Structural Analysis*).

Active Degrees of Freedom

Acoustic elements have acoustic pressure as a degree of freedom. Coupled acoustic-structural elements also have displacement degrees of freedom. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Choosing an Acoustic Element

Acoustic elements are available in the following element families:

- *Solid (Continuum) Elements*;
- *Infinite Elements*; and
- *Acoustic Interface Elements*.

The acoustic elements can be used alone but are often used with a structural model in a coupled analysis. *Acoustic Interface Elements* describes interface elements that allow this acoustic pressure field to be coupled to the displacements of the surface of the structure. Acoustic elements can also interact with solid elements through the use of surface-based tie constraints; see *Coupled Acoustic-Structural Analysis*.

Poroelastic Acoustic Elements

Volumetrically coupled poroelastic acoustic elements (or simply poroelastic elements) are used for modeling porous media undergoing small displacement and small pressure changes. The solution is defined by displacements and pressure variables. Interface boundary conditions connecting the poroelastic to poroelastic elements, poroelastic to elastic elements, and poroelastic to acoustic elements are available using surface-based tie constraints.

Analysis Types

Poroelastic elements are used in acoustic and coupled acoustic-structural analysis (*Coupled Acoustic-Structural Analysis*). They are available only in direct steady-state dynamic linear perturbation procedures.

Active Degrees of Freedom

Poroelastic elements have translational displacements and acoustic pressure degrees of freedom as primary variables. See *Conventions* for a discussion of the degrees of freedom in Abaqus.

Choosing a Poroelastic Acoustic Element

Poroelastic elements are available only in the following element family:

- *Solid (Continuum) Elements*.

Poroelastic elements can be used alone but are often used with a structural and acoustic model in a coupled analysis. Only first-order interpolation elements for both displacements and pressure are available.

Using the Same Mesh with Different Analysis or Element Types

You might want to use the same mesh with different analysis or element types. This might occur, for example, if both stress and heat transfer analyses are intended for a particular geometry or if the effect of using either reduced- or full-integration elements is being investigated. Care should be taken when doing this since unexpected error messages may result for one of the two element types if the mesh is distorted. For example, a stress analysis with C3D10 elements might run successfully, but a heat transfer analysis using the same mesh with DC3D10 elements might terminate during the **datacheck** portion of the analysis with an error message stating that the elements are excessively distorted or have negative volumes. This apparent inconsistency is caused by the different integration locations for the different element types. Such problems can be avoided by ensuring that the mesh is not distorted excessively.

Section Controls

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- **SECTION CONTROLS*
- **HOURGLASS STIFFNESS*
- *Element type assignment*

Overview

Section controls in Abaqus/Standard:

- choose the hourglass control formulation for most first-order elements with reduced integration;
- define the distortion control for C3D10HS elements;
- allow the activation as well as selection of the scale factor for bending stabilization in full-integration linear elements modeled with hyperelastic materials;
- select the hourglass control scale factors for all elements with reduced integration; and
- select the choice of element deletion and the value of maximum degradation for cohesive elements, connector elements, elements with plane stress formulations (plane stress, shell, continuum shell, and membrane elements) with constitutive behavior that includes damage evolution, any element that can be used with damage evolution models for ductile metals, and any element that can be used with the damage evolution law in a low-cycle fatigue analysis.

Section controls in Abaqus/Explicit:

- choose the hourglass control formulation or scale factors for all elements with reduced integration;
- define the distortion control for solid elements;
- select the scale factors for the drill stiffness of shell elements or deactivate the drill stiffness for small-strain shell elements S3RS and S4RS;
- select an amplitude for ramping of any initial stresses;
- select the kinematic formulation for hexahedron solid elements;
- select the accuracy order of the formulation for solid and shell elements;
- select the scale factors for linear and quadratic bulk viscosity parameters;
- specify the size of the particle tracking box for discrete element method (DEM) analyses and smoothed particle hydrodynamic (SPH) analyses;
- specify the formulation and additional control parameters for SPH analyses;
- select the choice of element deletion and the value of maximum degradation for elements with constitutive behavior that includes damage evolution;
- control the activation of the "improved" element time estimation method for three-dimensional continuum elements and elements with plane stress formulations;
- control shell element deletion based on integration point status;
- control distortion-based element deletion; and

- control the activation of bending stabilization for C3D8 elements modeled with hyperelastic materials and specify the scale factor.

In Abaqus/CAE section controls are specified when you assign an element type to particular mesh regions and are referred to as element controls.

Using Section Controls

In Abaqus/Standard section controls are used to select the enhanced hourglass control formulation for solid, shell, and membrane elements. This formulation provides improved coarse mesh accuracy with slightly higher computational cost and performs better for nonlinear material response at high strain levels when compared with the default total stiffness formulation. Section controls can also be used to select some element formulations that might be relevant for a subsequent Abaqus/Explicit analysis.

In Abaqus/Explicit the default formulations for solid, shell, and membrane elements have been chosen to perform satisfactorily on a wide class of quasi-static and explicit dynamic simulations. However, certain formulations give rise to some trade-off between accuracy and performance. Abaqus/Explicit provides section controls to modify these element formulations so that you can optimize these objectives for a specific application. Section controls can also be used in Abaqus/Explicit to specify scale factors for linear and quadratic bulk viscosity parameters. You can also control the initial stresses in membrane elements for applications such as airbags in crash simulations and introduce the initial stresses gradually based on an amplitude definition.

In addition, section controls are used to specify the maximum stiffness degradation and to choose the behavior on complete failure of an element, once the material stiffness is fully degraded, including the removal of failed elements from the mesh. This functionality applies only to elements with a material definition that includes progressive damage (see *About Progressive Damage and Failure*, *Connector Damage Behavior*, and *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*). In Abaqus/Standard this functionality is limited to

- cohesive elements with a traction-separation constitutive response that includes damage evolution,
- any element with a plane stress formulation that can be used with the damage evolution model for fiber-reinforced composites,
- any element that can be used with the damage evolution models for ductile metals,
- any element that can be used with the damage evolution law in a low-cycle fatigue analysis, and
- connector elements with a constitutive response that includes damage evolution.

Input File Usage: Use the following option to specify a section controls definition:

**SECTION CONTROLS*, NAME=*name*

This option is used in conjunction with one or more of the following options to associate the section control definition with an element section definition:

**COHESIVE SECTION*, CONTROLS=*name*
**CONNECTOR SECTION*, CONTROLS=*name*
**DISCRETE SECTION*, CONTROLS=*name*
**EULERIAN SECTION*, CONTROLS=*name*
**MEMBRANE SECTION*, CONTROLS=*name*
**SHELL GENERAL SECTION*, CONTROLS=*name*
**SHELL SECTION*, CONTROLS=*name*
**SOLID SECTION*, CONTROLS=*name*

You can apply a single section control definition to several element section definitions.

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Element Controls**

Methods for Suppressing Hourglass Modes

The formulation for reduced-integration elements considers only the linearly varying part of the incremental displacement field in the element for the calculation of the increment of physical strain. The remaining part of the nodal incremental displacement field is the hourglass field and can be expressed in terms of hourglass modes.

Excitation of these modes might lead to severe mesh distortion, with no stresses resisting the deformation. Similarly, the formulation for element type C3D4H considers in the constraint equations only the constant part of the incremental pressure Lagrange multiplier field. The remaining part of the nodal incremental pressure Lagrange multiplier interpolation is comprised of hourglass modes.

Hourglass control attempts to minimize these problems without introducing excessive constraints on the element's physical response.

Several methods are available in Abaqus for suppressing the hourglass modes, as described below.

Integral Viscoelastic Approach in Abaqus/Explicit

The integral viscoelastic approach available in Abaqus/Explicit generates more resistance to hourglass forces early in the analysis step where sudden dynamic loading is more probable.

Let q be an hourglass mode magnitude and Q be the force (or moment) conjugate to q. The integral viscoelastic approach is defined as

$$Q = \int_0^t sK\left(t - t'\right)\frac{dq}{dt'}dt',$$

where K is the hourglass stiffness selected by Abaqus/Explicit, and s is one of up to three scaling factors s^s, s^r, and s^w that you can define (by default, $s^s = s^r = s^w = 1.0$). The scale factors are dimensionless and relate to specific displacement degrees of freedom. For solid elements s^s scales all hourglass stiffnesses. For membrane elements s^s scales the hourglass stiffnesses related to the in-plane displacement degrees of freedom, and s^w scales the out-of-plane displacement degrees of freedom. For shell elements s^s scales the hourglass stiffnesses related to the in-plane displacement degrees of freedom, and s^r scales the hourglass stiffnesses related to the rotational degrees of freedom. In addition, s^w scales the hourglass stiffness related to the transverse displacement for small-strain shell elements.

The integral viscoelastic form of hourglass control is available for all reduced-integration elements and is the default form in Abaqus/Explicit, except for elements modeled with hyperelastic, hyperfoam, and low-density foam materials and for Eulerian EC3D8R elements. It is the most computationally intensive hourglass control method. It is not supported for Eulerian EC3D8R elements.

Input File Usage: **SECTION CONTROLS*, NAME=*name*,
HOURGLASS=RELAX STIFFNESS
s^s, s^r, s^w

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Hourglass control: Relax stiffness**, **Displacement hourglass scaling factor:** s^s, **Rotational hourglass scaling factor:** s^r, **Out-of-plane displacement hourglass scaling factor:** s^w

Kelvin Viscoelastic Approach in Abaqus/Explicit

The Kelvin-type viscoelastic approach available in Abaqus/Explicit is defined as

$$Q = s\left[(1-\alpha)Kq + \alpha C\frac{dq}{dt}\right],$$

where K is the linear stiffness and C is the linear viscous coefficient. This general form has pure stiffness and pure viscous hourglass control as limiting cases. When the combination is used, the stiffness term acts to maintain a nominal resistance to hourglassing throughout the simulation and the viscous term generates additional resistance to hourglassing under dynamic loading conditions.

Three approaches are provided in Abaqus/Explicit for specifying Kelvin viscoelastic hourglass control.

Specifying the Pure Stiffness Approach

The pure stiffness form of hourglass control is available for all reduced-integration elements and is recommended for both quasi-static and transient dynamic simulations.

Input File Usage: **SECTION CONTROLS*, NAME=*name*, HOURGLASS=STIFFNESS
s^s, s^r, s^w

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Hourglass control: Stiffness**, **Displacement hourglass scaling factor: s^s**, **Rotational hourglass scaling factor: s^r**, **Out-of-plane displacement hourglass scaling factor: s^w**

Specifying the Pure Viscous Approach

The pure viscous form of hourglass control is available only for solid and membrane elements with reduced integration and is the default form in Abaqus/Explicit for Eulerian EC3D8R elements. It is the most computationally efficient form of hourglass control and has been shown to be effective for high-rate dynamic simulations. However, the pure viscous method is not recommended for low frequency dynamic or quasi-static problems since continuous (static) loading in hourglass modes results in excessive hourglass deformation due to the lack of any nominal stiffness.

Input File Usage: **SECTION CONTROLS*, NAME=*name*, HOURGLASS=VISCOUS
s^s, s^r, s^w

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Hourglass control: Viscous**, **Displacement hourglass scaling factor: s^s**, **Rotational hourglass scaling factor: s^r**, **Out-of-plane displacement hourglass scaling factor: s^w**

Specifying a Combination of Stiffness and Viscous Hourglass Control

A linear combination of stiffness and viscous hourglass control is available only for solid and membrane elements with reduced integration. You can specify the blending weight factor α ($0 \le \alpha \le 1$) to scale the stiffness and viscous contributions. Specifying a weight factor equal to 0.0 or 1.0 results in the limiting cases of pure stiffness and pure viscous hourglass control, respectively. The default weight factor is 0.5.

Input File Usage: **SECTION CONTROLS*, NAME=*name*, HOURGLASS=COMBINED,
WEIGHT FACTOR=α
s^s, s^r, s^w

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Hourglass control: Combined**, **Stiffness-viscous weight factor:** α, **Displacement hourglass scaling factor:** s^s, **Rotational hourglass scaling factor:** s^r, **Out-of-plane displacement hourglass scaling factor:** s^w

Total Stiffness Approach in Abaqus/Standard

The total stiffness approach available in Abaqus/Standard is the default hourglass control approach for all first-order, reduced-integration elements in Abaqus/Standard, except for elements modeled with hyperelastic, hyperfoam, or hysteresis materials. It is the only hourglass control approach available in Abaqus/Standard for S8R5, S9R5, and M3D9R elements and the only hourglass control approach available for the pressure Lagrange multiplier interpolation for C3D4H elements. Hourglass stiffness factors for first-order, reduced-integration elements depend on the shear modulus, while factors for C3D4H elements depend on the bulk modulus. A scale factor can be applied to these stiffness factors to increase or decrease the hourglass stiffness.

Let q be an hourglass mode magnitude and Q be the force (moment, pressure, or volumetric flux) conjugate to q. The total stiffness approach for hourglass control in membrane or solid elements or membrane hourglass control in shell elements is defined as

$$Q = s^s \left((r_F G)\ B_\alpha^P B_\alpha^P\ V \right) q,$$

where s^s is a dimensionless scale factor (by default, $s^s = 1.0$); $r_F G$ is an hourglass stiffness factor with units of stress; B_α^P is the gradient interpolator used to define constant gradients in the element ($\partial u / \partial S_\alpha = B_\alpha^P u^P$, where the superscript P refers to an element node, the subscript α refers to a direction, and S_α is a material coordinate); and V is the element volume. Similarly, the hourglass control for the pressure Lagrange multiplier interpolation for C3D4H elements is defined as

$$Q = s^p \left((r_F K)\ B^P B^P\ V \right) q,$$

where s^p is a dimensionless scale factor (by default, $s^p = 1.0$); B^P is a volumetric gradient operator; and $r_F K$ is an hourglass stiffness factor with units of stress for compressible hyperelastic and hyperfoam materials and units of stress compliance for all other materials. The total stiffness approach for bending hourglass control in shell elements is defined as

$$Q = s^r \left((r_\theta G)\ B_\alpha^P B_\alpha^P\ t^3 A \right) q,$$

where s^r is the scale factor (by default, $s^r = 1.0$), $r_\theta G$ is the hourglass stiffness factor, t is the thickness of the shell element, and A is the area of the shell element.

Input File Usage: **SECTION CONTROLS*, NAME=*name*, HOURGLASS=STIFFNESS
s^s, s^r, , , , s^p

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Hourglass control: Stiffness**, **Displacement hourglass scaling factor:** s^s, **Rotational hourglass scaling factor:** s^r

Default Hourglass Stiffness Values

Normally the hourglass control stiffness is defined from the elasticity associated with the material. In most cases, the control stiffness of first-order, reduced-integration elements is based on a typical value of the initial shear modulus of the material, which might, for example, be given as part of the elastic material definition (*Linear Elastic Behavior*). Similarly, hourglass control stiffness of the reduced-integration pressure and volumetric Lagrange multiplier interpolations of C3D4H elements is based on a typical value of the initial bulk modulus. For an isotropic elastic or hyperelastic material G is the shear modulus. For a nonisotropic elastic material average moduli are used to calculate the hourglass stiffness: for orthotropic elasticity defined by specifying the terms in the elastic stiffness matrix or for anisotropic elasticity

$$G = \frac{1}{3}\left(D_{1212} + D_{1313} + D_{2323}\right)$$

and for orthotropic elasticity defined by specifying the engineering constants or for orthotropic elasticity in plane stress

$$G = \frac{1}{3}\left(G_{12} + G_{13} + G_{23}\right).$$

If the elastic moduli are dependent on temperature or field variables, the first value in the table is used. The default values for the stiffness factors are defined below.

For membrane or solid elements

$$(r_F G) = 0.005\, G.$$

For membrane hourglass control in a shell

$$(r_F G) = 0.005\, \frac{\int_{-t/2}^{t/2} G\, dt}{t}.$$

For control of bending hourglass modes in a shell

$$(r_\theta G) = 0.00375\, \frac{12 \int_{-t/2}^{t/2} G t^2\, dt}{t^3}.$$

For a general shell section defined by specifying the equivalent section properties directly, t is defined as

$$t = \sqrt{12 \frac{D_{44} + D_{55} + D_{66}}{D_{11} + D_{22} + D_{33}}}$$

and an effective shear modulus for the section is used to calculate the hourglass stiffness:

$$G = \frac{1}{6t}\left(D_{11} + D_{22}\right) + \frac{1}{3t} D_{33},$$

where D_{ij} is the section stiffness matrix.

User-Defined Hourglass Stiffness

When the initial shear modulus is not defined, you must define the hourglass stiffness parameters; an example is when user subroutine *UMAT* is used to describe the material behavior of elements with hourglassing modes. In some cases the default value provided for the hourglass control stiffness might not be suitable and you should define the value.

In some coupled pore fluid diffusion and stress analyses the prevailing pore pressure in the medium might approach the magnitude of the stiffness of the material skeleton, as measured by constitutive parameters such as the elastic modulus. These cases are expected in some partial saturation evaluations of the wetting of relatively compliant materials such as tissues or cloth. When reduced-integration or modified tetrahedral or triangular elements are used in such analyses, the default choice of the hourglass control stiffness parameter, which is based

on a scaling of skeleton material constitutive parameters, might not be adequate to control hourglassing in the presence of large pore pressure fields. An appropriate hourglass control stiffness in these cases should scale with the expected magnitude of pore pressure changes over an element.

Input File Usage: Use the following option to specify nondefault values for the hourglass stiffness factors:

**HOURGLASS STIFFNESS*

$r_F G$, $r_F K$, $r_\theta G$, *drilling hourglass scaling factor for shells*

This option must immediately follow one of the following options:

**MEMBRANE SECTION*
**SHELL GENERAL SECTION*
**SHELL SECTION*
**SOLID SECTION*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Hourglass stiffness: Specify** $r_F G$ or for shells **Membrane hourglass stiffness: Specify** $r_F G$, **Bending hourglass stiffness: Specify** $r_\theta G$, and **Drilling hourglass scaling factor: Specify** *drilling hourglass scaling factor for shells*

Enhanced Hourglass Control Approach in Abaqus/Standard and Abaqus/Explicit

The enhanced hourglass control approach available in both Abaqus/Standard and Abaqus/Explicit represents a refinement of the pure stiffness method in which the stiffness coefficients are based on the enhanced assumed strain method; no scale factor is required. It is the default hourglass control approach for hyperelastic, hyperfoam, and low-density foam materials in Abaqus/Explicit and for hyperelastic, hyperfoam, and hysteresis materials in Abaqus/Standard. This method gives more accurate displacement solutions for coarse meshes with linear elastic materials as compared to other hourglass control methods. It also provides increased resistance to hourglassing for nonlinear materials. Although generally beneficial, this might give overly stiff response in problems displaying plastic yielding under bending. In Abaqus/Explicit the enhanced hourglass method will generally predict a much better return to the original configuration for hyperelastic or hyperfoam materials when loading is removed.

The enhanced hourglass control approach is compatible between Abaqus/Standard and Abaqus/Explicit. It is recommended that enhanced hourglass control be used for both Abaqus/Standard and Abaqus/Explicit for all import analyses. See *Transferring Results between Abaqus/Explicit and Abaqus/Standard*.

The enhanced hourglass method is not supported for enriched elements (see *Modeling Discontinuities as an Enriched Feature Using the Extended Finite Element Method*).

Specifying the Enhanced Hourglass Control Approach

The enhanced hourglass control method is available for first-order solid, membrane, and finite-strain shell elements with reduced integration. In Abaqus/Explicit it cannot be used for a hyperelastic or hyperfoam material when adaptive meshing is used on that domain (see the discussion below).

Input File Usage: **SECTION CONTROLS*, NAME=*name*, HOURGLASS=ENHANCED

Any scaling factors specified on the data line following this option will be ignored.

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Hourglass control: Enhanced**

Special Considerations for Hyperelastic and Hyperfoam Materials in an Adaptive Mesh Domain in Abaqus/Explicit

The enhanced hourglass method cannot be used with elements modeled with hyperelastic or hyperfoam materials that are included in an adaptive mesh domain. Thus, if you decide to use hyperelastic or hyperfoam materials in an adaptive mesh domain, you must specify section controls to choose a different hourglass control approach. The use of adaptive meshing in domains modeled with finite-strain elastic materials is not recommended since better results are generally predicted using the enhanced hourglass method and, for solid elements, element distortion control (discussed below). Therefore, for these materials it is recommended that the analysis be run without adaptive meshing but with enhanced hourglass control.

Use in Coupled Pore Pressure Analysis

When first-order, reduced-integration, or modified tetrahedral or triangular elements are used in coupled pore fluid diffusion and stress analyses or coupled temperature–pore pressure analyses with enhanced hourglass control, the hourglass control stiffness, which is based on skeleton material constitutive parameters, might not be adequate to control hourglassing in the presence of large pore pressure fields. Since enhanced hourglass control does not allow you to change the hourglass control stiffness, it is recommended that total stiffness hourglass control be used in these cases with an appropriate hourglass control stiffness scaled with the expected magnitude of pore pressure changes over an element.

Controlling Element Distortion for Crushable Materials in Abaqus/Explicit

Many analyses with volumetrically compacting materials such as crushable foams see large compressive and shear deformations, especially when the crushable materials are used as energy absorbers between stiff or heavy components. The material behavior for crushable materials usually stiffens significantly under high compression. When a finer mesh is used, the stiffening behavior of the material model is enough to prevent negative element volumes or other excessive distortion from occurring under high compressive loads. However, analyses might fail prematurely when the mesh is coarse relative to strain gradients and the amount of compression.

Abaqus/Explicit offers distortion control to prevent solid elements from inverting or distorting excessively for these cases. The constraint method used in Abaqus/Explicit prevents each node on an element from punching inward toward the center of the element past a point where the element would become nonconvex. Constraints are enforced by using a penalty approach, and you can control the associated distortion length ratio.

Distortion control is available only for solid elements and cannot be used when the elements are included in an adaptive mesh domain. Distortion control is activated by default for elements modeled with hyperelastic, hyperfoam, crushable foam, or low-density foam materials when the default hourglass control method is used. If you decide to use any of these materials with solid elements included in an adaptive mesh domain, you must specify section controls to deactivate distortion control. Using adaptive meshing in a domain modeled with hyperelastic or hyperfoam materials is not recommended since better results are generally predicted using the enhanced hourglass method in combination with element distortion control.

When element distortion control is used in combination with the enhanced hourglass method (default behavior for hyperelastic and hyperfoam materials), a small amount of viscous damping is added to the element formulation and the associated viscous energy dissipation is included in the output of artificial strain energy (ALLAE).

If distortion control is used, the energy dissipated by distortion control can be output on request (see *Abaqus/Explicit Output Variable Identifiers* for details). Although developed for analyses of energy absorbing, volumetrically compacting materials, distortion control can be used with any material model. However, care must be used in interpreting results since the distortion control constraints might inhibit legitimate deformation modes and lock up the mesh. Distortion control cannot prevent elements from being distorted due to temporal instabilities, hourglass instabilities, or physically unrealistic deformation.

Input File Usage: Use the following option to activate distortion control:

**SECTION CONTROLS*, NAME=*name*, DISTORTION CONTROL=YES

Use the following option to deactivate distortion control:

**SECTION CONTROLS*, NAME=*name*, DISTORTION CONTROL=NO

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Distortion control: Yes** or **No**

Controlling the Distortion Length Ratio

By default, the constraint penalty forces are applied when the node moves to a point a small offset distance away from the actual plane of constraint. This appears to improve the robustness of the method and limits the reduction of time increment due to severe shortening of the element characteristic length. This offset distance is determined by the distortion length ratio times the initial element characteristic length. The default value of the distortion length ratio, *r*, is 0.1. You can change the distortion length ratio by specifying a value for *r*, $0 < r \leq 1$.

For C3D10 elements, instead of the distortion length ratio, the distortion volume ratio (the current volume over the original volume at each integration point) is used. The default threshold value of the distortion volume ratio, *r*, is also 0.1. You can change the distortion volume ratio by specifying a value for *r*, $0 < r \leq 1$.

Input File Usage: **SECTION CONTROLS*, NAME=*name*, DISTORTION CONTROL=YES, LENGTH RATIO=*r*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Distortion control: Yes**, **Length ratio:** *r*

Selecting a Scale Factor for the Drill Stiffness in Abaqus/Explicit

A drill constraint acts to keep the element nodal rotations in the direction of the shell normal consistent with the average in-plane rotation of the element. Lack of such a constraint can lead to large rotations at these element nodes. Section controls can be used to select a scale factor for the default drill stiffness of an individual element set.

Input File Usage: Use the following options to specify a scale factor for the drill stiffness:

**SECTION CONTROLS*, NAME=*name*
, , , , , , , *scale factor for drill stiffness*

Drill Constraint in Small Strain Shell Elements S3RS and S4RS in Abaqus/Explicit

The formulation of small strain shell elements S3RS and S4RS includes a drill constraint and does so by default. Alternatively, you can deactivate the drill constraint for these elements. The drill constraint is always active for the finite strain conventional shell elements such as S4R, but the default value of the drill stiffness can be scaled as mentioned above.

Input File Usage: Use the following option to activate the drill constraint (default):

**SECTION CONTROLS*, DRILL STIFFNESS=ON

Use the following option to deactivate the drill constraint:

**SECTION CONTROLS*, DRILL STIFFNESS=OFF

Ramping of Initial Stresses in Abaqus/Explicit

Abaqus/Explicit provides a technique to introduce initial stresses gradually based on an amplitude definition to reduce possible oscillations in the initial solution. You must define this amplitude with a value starting from zero and reaching a final value of one. The initial stresses are not applied for the duration that the amplitude stays at zero, and the incurred energy for the duration that the amplitude is greater than zero is added to the system through output variable ALLVD. This technique is available with solid elements, membrane elements, beam elements, and shell elements (S4R and S4RS).

You can use the ramping technique in combination with user-defined initial stresses (see *Defining Initial Stresses*) or together with a reference mesh definition (see *Defining a Reference Mesh for Membrane Elements and Three-Dimensional Solid Elements*).

In crash-related simulations (such as airbag deployment or dummy positioning/settling), the initial stresses are often introduced in the model through the definition of a reference configuration that is different from the initial configuration. For airbag modeling, the components that confine the airbag in the initial configuration are often excluded from the numerical model, which might cause motion of the airbag under initial stresses at the beginning of the analysis. For dummy positioning/settling modeling, the initial stresses introduced by the reference mesh might cause initial oscillations in the solution. By ramping the initial stresses, you can avoid or reduce the initial motion of the airbag and the initial oscillations.

Input File Usage: Use both of the following options:

**AMPLITUDE*, NAME=*name*
**SECTION CONTROLS*, RAMP INITIAL STRESS=*name*

Defining the Kinematic Formulation for Hexahedron Solid Elements

The default kinematic formulation for reduced-integration solid elements in Abaqus (and the only kinematic formulation available in Abaqus/Standard) is based on the uniform strain operator and the hourglass shape vectors. Details can be found in *Solid isoparametric quadrilaterals and hexahedra*. These kinematic assumptions result in elements that pass the constant strain patch test for a general configuration and give zero strain under large rigid body rotation. However, the formulation is relatively expensive, especially in three dimensions.

Abaqus/Explicit offers two alternative kinematic formulations for the C3D8R solid element that can reduce the computational cost. The performance for each kinematic formulation on the patch test and under large rigid body rotation for various element configurations is summarized in *Table 1*. Suitable applications for each kinematic formulation are summarized in *Table 2*.

Table 1: Element performance for patch test and large rigid body rotations for various element configurations.

	Element configuration	Kinematic formulation type		
		Average strain	Orthogonal	Centroid
Satisfaction of the three-dimensional patch test	Parallelepiped	Yes	Yes	Yes
	General	Yes	No	No
Zero straining under rigid body rotation	Parallelepiped	Yes	Yes	Yes
	General	Yes	Yes	No

Table 2: Different element formulations and their suitable applications. The default formulation is highlighted below.

Kinematic formulation	Order of accuracy	Suitable applications
Average strain	Second-order	All; recommended for problems involving a large number of revolutions (>5).
Average strain	First-order	All; except those involving a large number of revolutions (>5).
Orthogonal	—	All; except those involving high confinement, very coarse meshes, or highly distorted elements.
Centroid	—	Problems with little rigid body rotation and reasonable mesh refinement.

You can specify the kinematic formulation for 8-node brick elements.

Default Formulation

The default average strain formulation of uniform strain and hourglass shape vectors is the only formulation available in Abaqus/Standard. This formulation is recommended for all problems and is particularly well suited for applications exhibiting high confinement, such as closed-die forming and bushing analyses.

Input File Usage: **SECTION CONTROLS*, KINEMATIC SPLIT=AVERAGE STRAIN

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Kinematic split: Average strain**

Orthogonal Formulation in Abaqus/Explicit

A noticeable reduction in computational cost can be obtained by using the orthogonal formulation available in Abaqus/Explicit. This formulation is based on the centroidal strain operator and a slight modification to the hourglass shape vectors. The centroidal strain operator requires three times fewer floating point operations than the uniform strain operator. Elements formulated with an orthogonal kinematic split pass the patch test only for rectangular or parallelepiped element configurations. However, numerical experience has shown that the element converges on the exact solution for general element configurations as the mesh is refined. It also performs well for large rigid body motions.

This formulation provides a good balance between computational speed and accuracy. It is recommended for all analyses except those involving highly distorted elements, very coarse meshes, or high confinement. Suitable applications for this formulation include elastic drop testing.

Input File Usage: **SECTION CONTROLS*, NAME=*name*,
KINEMATIC SPLIT=ORTHOGONAL

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Kinematic split: Orthogonal**

Centroid Formulation in Abaqus/Explicit

The fastest formulation available in Abaqus/Explicit is specified by selecting the centroid formulation. The centroid formulation is based on the centroidal strain operator and the hourglass base vectors. Using the hourglass base vectors instead of the hourglass shape vectors reduces hourglass mode computations by a factor of three. However, the hourglass base vectors are not orthogonal to rigid body rotation for general element configurations, so that hourglass strain might be generated with large rigid body rotations with this formulation.

This formulation should be used only to improve computational performance on problems that have reasonable mesh refinement and no significant amount of rigid body rotation (for example, transient flat rolling simulation).

Input File Usage: **SECTION CONTROLS*, NAME=*name*, KINEMATIC SPLIT=CENTROID

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Kinematic split: Centroid**

Choosing the Order of Accuracy in Solid and Shell Element Formulations

Abaqus/Standard offers only a second-order accurate formulation for all elements.

Abaqus/Explicit offers both first- and second-order accurate formulations for solid and shell elements. First-order accuracy is the default and yields sufficient accuracy for nearly all Abaqus/Explicit problems because of the inherently small time increment size. Second-order accuracy is usually required for analyses with components undergoing a large number of revolutions (>5). For three-dimensional solids the second-order accuracy formulation is available only with the default average strain kinematic formulation.

First-Order Accuracy

In Abaqus/Explicit the first-order accurate formulation for solid and shell elements is the default. This formulation is not available in Abaqus/Standard.

Input File Usage: **SECTION CONTROLS*, NAME=*name*,
SECOND ORDER ACCURACY=NO

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Second-order accuracy: No**

Second-Order Accuracy

The second-order accurate element formulation is appropriate for problems with a large number of revolutions (>5). This is the only formulation available in Abaqus/Standard. *Simulation of propeller rotation* illustrates the performance of second-order accurate shell and solid elements in Abaqus/Explicit as they undergo about 100 revolutions.

Input File Usage: **SECTION CONTROLS*, NAME=*name*,
SECOND ORDER ACCURACY=YES

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Second-order accuracy: Yes**

Selecting Scale Factors for Bulk Viscosity in Abaqus/Explicit

Bulk viscosity introduces damping associated with volumetric straining. Its purpose is to improve the modeling of high-speed dynamic events. Abaqus/Explicit contains two forms of bulk viscosity, linear and quadratic, which can be defined for the whole model at each step of the analysis, as discussed in *Bulk Viscosity*. Section controls can be used to select scale factors for the linear and quadratic bulk viscosities of an individual element set.

The pressure term generated by bulk viscosity might introduce unexpected results in the volumetric response of highly compressible materials; therefore, it is recommended to suppress bulk viscosity for these materials by specifying scale factors equal to zero.

Input File Usage: Use the following options to specify scale factors for the linear and quadratic bulk viscosities:

**SECTION CONTROLS*, NAME=*name*
, , , *scale factor for linear bulk viscosity, scale factor for quadratic bulk viscosity*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Linear bulk viscosity scaling factor** or **Quadratic bulk viscosity scaling factor**

Controlling the Activation of the "Improved" Element Time Estimation Method in Abaqus/Explicit

For three-dimensional continuum elements and elements with plane stress formulations (shell, membrane, and two-dimensional plane stress elements) an "improved" estimate of the element characteristic length is used by default. This "improved" method usually results in a larger element stable time increment than a more traditional method. The activation of the "improved" element time estimation method can be defined globally for the whole model at each step of the analysis, as discussed in *Time Incrementation*. Alternatively, you can selectively control the activation of the "improved" element time estimation method for each individual element set.

Input File Usage: Use the following option to match the setting of the "improved" element time estimation method defined globally for the whole model:

**SECTION CONTROLS*, NAME=*name*,
IMPROVED DT METHOD=GLOBAL

Use the following option to activate the "improved" element time estimation method:

**SECTION CONTROLS*, NAME=*name*,
IMPROVED DT METHOD=YES

Use the following option to deactivate the "improved" element time estimation method:

**SECTION CONTROLS*, NAME=*name*,
IMPROVED DT METHOD=NO

Abaqus/CAE Usage: Controlling the activation of the "improved" element time estimation method is not supported in Abaqus/CAE.

Controlling Element Deletion and Maximum Degradation for Materials with Damage Evolution

Abaqus offers a general capability for modeling progressive damage and failure of materials (*About Progressive Damage and Failure*). In Abaqus/Standard this capability is available only for cohesive elements, connector elements, elements with plane stress formulations (plane stress, shell, continuum shell, and membrane elements), any element that can be used with the multiscale damage evolution models for fiber-reinforced composites, any element that can be used with the damage evolution models for ductile metals, and any element that can be used with the damage evolution law in a low-cycle fatigue analysis. In Abaqus/Explicit this capability is available for all elements with progressive damage behavior except connector elements. Section controls are provided to specify the value of the maximum stiffness degradation, D_{max}, for material failure and whether element deletion occurs when the degradation reaches this level. By default, an element is deleted on material failure, except for

elements used with the multiscale damage evolution models for fiber-reinforced composites. Details for element deletion driven by material failure are described in *Material Failure and Element Deletion*. The choice of element deletion also affects how the damage is applied; details can be found in the following sections:

- *Maximum Degradation and Choice of Element Removal*;
- *Maximum Degradation and Choice of Element Removal in Abaqus/Standard*;
- *Maximum Degradation and Choice of Element Removal*;
- *Maximum Degradation and Choice of Element Removal*; and
- *Damage Evolution for Ductile Materials in Low-Cycle Fatigue*.

Input File Usage: Use the following option to delete the element from the mesh:

**SECTION CONTROLS*, ELEMENT DELETION=YES

Use the following option to keep the element in the computation:

**SECTION CONTROLS*, ELEMENT DELETION=NO

Use the following option to specify $\boldsymbol{D}_{\mathbf{max}}$:

**SECTION CONTROLS*, MAX DEGRADATION=$\boldsymbol{D}_{\mathbf{max}}$

Abaqus/CAE Usage: Use the following option to control whether completely damaged elements remain in the computation:

Mesh module: **Mesh->Element Type**: **Element deletion**

Use the following option to determine when an element is considered completely damaged:

Mesh module: **Mesh->Element Type**: **Max degradation**

Controlling Shell Element Deletion Based on Integration Point Status in Abaqus/Explicit

In Abaqus/Explicit you can delete shell elements based on the number of active or failed integration points through the shell section. By default, when only one integration point through the shell section is active, the shell element is deleted. You can specify either the number of active integration points or the number of failed integration points through the shell section at which the shell element is deleted. The number you specify must not exceed the total number of integration points through the shell section. Alternatively, you can specify that a shell element is deleted when all the integration points through the shell section are failed.

Input File Usage: Use the following option to specify the number of active integration points through the shell section at which the shell element is deleted:

**SECTION CONTROLS*, SHELL DELETION NUMBER=*positive integer*

Use the following option to specify the number of failed integration points through the shell section at which the shell element is deleted:

**SECTION CONTROLS*, SHELL DELETION NUMBER=*negative integer*

Use the following option to delete the shell element when all the integration points through the shell section are failed:

*SECTION CONTROLS, SHELL DELETION NUMBER=0

Abaqus/CAE Usage: Controlling shell element deletion based on integration point status is not supported in Abaqus/CAE.

Controlling Distortion-Based Element Deletion in Abaqus/Explicit

In Abaqus/Explicit you can control element deletion for deformable elements (except cohesive elements) based on distortion. An inherent limit exists as to how much deformation a Lagrangian element can accommodate to obtain accurate simulation results. By default, elements are deleted only when material failure is defined and failure occurs. In rare cases, an element can become excessively distorted before material failure occurs and cause the simulation to stop prematurely. In addition, when an element gets distorted, the element stable time increment can drop dramatically, causing a performance issue. You can activate distortion-based element deletion to delete the distorted elements and allow the simulation to continue.

You can control the deletion of distorted elements using the following measures: current element stable time increment, volume or area, or characteristic length. The element stable time increment used does not include damping (bulk viscosity), mass scaling, or penalty contact effects. The element is removed once the measure falls below the specified value. The deletion criteria can be based on a ratio of the deletion measure over the original value; for example, the ratio of the element volume over the original element volume at which the element is deleted. Alternatively, you can specify a value for the deletion measure; for example, the element volume at which the element is deleted.

Distortion-based element deletion is intended for experienced users and should be used with caution. Setting improper values for the deletion criteria can lead to unphysical or misleading results.

Input File Usage: Use the following option to activate distortion-based element deletion and to specify the associated distortion criteria:

*SECTION CONTROLS, DELETE DISTORTED ELEMENT=YES
first data line
second data line
third data line
fourth data line
fifth data line
stable time increment ratio, volume or area ratio, characteristic length ratio, stable time increment,
volume or area, characteristic length

Abaqus/CAE Usage: Controlling distortion-based element deletion is not supported in Abaqus/CAE.

Using Linear Kinematic Conversion in Abaqus/Explicit

When elements are subject to large compressive forces, they can reach a point when negative volumes are calculated. The negative volumes cause a fatal error if nonlinear geometric effects are considered. One method to avoid the fatal error is to convert the element from nonlinear kinematics to linear kinematics when it reaches a certain level of compression (volume reduction).

Linear kinematic conversion is activated only when distortion control is also activated. The element is converted when distortion control forces become active. For C3D10 elements, linear kinematic conversion does not require the use of distortion control. For C3D10 elements, linear kinematic conversion occurs when the ratio of the element characteristic length over the original characteristic length reaches a critical value.

Linear kinematic conversion allows the simulation to continue subject to approximation of linear kinematics for the highly distorted elements. Linear kinematic conversion can be used as an alternative to distortion-based element deletion (see *Controlling Distortion-Based Element Deletion in Abaqus/Explicit*).

The linear kinematic conversion is mainly developed to improve element robustness. The element computation results will not be exactly accurate for the elements that have been converted to linear kinematics. Therefore, linear kinematic conversion should be used with caution to avoid unphysical or misleading results.

Input File Usage: Use the following option to activate linear kinematic conversion:

**SECTION CONTROLS*, LINEAR KINEMATIC CONVERSION

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Linear kinematic conversion**

Using Viscous Regularization with Cohesive Elements, Connector Elements, and Elements That Can Be Used with the Damage Evolution Models for Ductile Metals and Fiber-Reinforced Composites Including the Multiscale Model for Fiber-Reinforced Composites in Abaqus/Standard

Material models exhibiting softening behavior and stiffness degradation often lead to severe convergence difficulties in implicit analysis programs, such as Abaqus/Standard. A common technique to overcome some of these convergence difficulties is the use of viscous regularization of the constitutive equations, which causes the tangent stiffness matrix of the softening material to be positive for sufficiently small time increments.

The traction-separation laws used to describe the constitutive behavior of cohesive elements can be regularized in Abaqus/Standard using viscosity, by permitting stresses to be outside the limits defined by the traction-separation law. The details of the regularization procedure are discussed in *Viscous Regularization in Abaqus/Standard*. The same technique is also used to regularize the following:

- damaged (softening) connector response (see *Connector Damage Behavior*),
- damaged response of elements with plane stress formulations when they are used with the damage model for fiber-reinforced materials (see *Viscous Regularization*), and
- damage response of elements used with the damage model for ductile metals (see *Damage Evolution and Element Removal for Ductile Metals*).

You specify the amount of viscosity to be used for the regularization procedure. By default, no viscosity is included so that no viscous regularization is performed.

Input File Usage: **SECTION CONTROLS*, VISCOSITY=μ

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Viscosity**

Using Section Controls for Slurry Transport in Abaqus/Standard

In slurry transport analysis in Abaqus/Standard, slurry concentration is represented by the volumetric fraction of particles in a homogenized way. Although the particles are not tracked explicitly, you can control if particles are allowed to move into a fracture by specifying a minimum ratio, α, between the crack width and particle diameter. Particles can only move into a fracture with a gap opening of at least α times the particle diameter. By default, the minimum ratio α is 1.0.

Input File Usage: **SECTION CONTROLS*, MIN GAP PARTICLE RATIO=α

Abaqus/CAE Usage: You cannot specify a minimum gap particle ratio in Abaqus/CAE.

Using Viscous Damping with Connector Elements in Abaqus/Standard

Material failure in connector elements often causes convergence problems in Abaqus/Standard. To avoid such convergence problems, you can introduce viscous damping into the connector components by specifying the value of the damping coefficient as discussed in *Connector Failure Behavior*. By default, no damping is included.

Input File Usage: **SECTION CONTROLS*, VISCOSITY=μ

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Viscosity**

Using Section Controls in an Import Analysis

The recommended procedure for doing import analysis is to specify the enhanced hourglass control formulation in the original analysis. Once the section controls have been specified in the original analysis, they cannot be modified in subsequent import analyses. This ensures that the enhanced hourglass control formulation is used in the original as well as import analyses. The default values for other section controls are usually appropriate and should not be changed. For further details on using section controls in an import analysis, see *Transferring Results between Abaqus/Explicit and Abaqus/Standard*.

Using Section Controls for Flexion-Torsion Type Connector

When the third axes of the two local coordinate systems for a flexion-torsion type connector are exactly aligned, a numerical singularity occurs that might lead to convergence difficulties. To avoid this, a small perturbation can be applied to the local coordinate system defined at the second connector node.

Input File Usage: **SECTION CONTROLS*, PERTURBATION=*small angle*

Abaqus/CAE Usage: You cannot specify a perturbation for flexion-torsion type connectors in Abaqus/CAE.

Using Section Controls to Define the Particle Tracking Box for DEM and SPH Particles

For discrete element method (DEM) analyses, a particle tracking box is established at the beginning of the analysis to define the rectangular region within which the particle search (finding all neighbors for all particles) is performed. A region that is 10% larger in all directions than the overall model initial dimensions and is centered at the geometric center of the model is used.

For smoothed particle hydrodynamic (SPH) analyses, all particles are tracked as the analysis progresses by default. For DEM analyses, particle tracking is based on the initially established tracking box by default. Alternatively, you can define a particle tracking box to define the region within which the particle search is performed.

You define a fixed size for the particle tracking box by specifying the coordinates of two opposite corners (lower left and upper right) of this box. As the analysis progresses, if a particle is outside this tracking box, it behaves like a free-flying point mass and does not contribute to the DEM or SPH calculations. If the particle reenters the box at a later stage, it is once again included in the calculations. If you want to track all the particles during the analysis, you must ensure that the particle tracking box fully encompasses the domain through which the model moves; otherwise, you will lose tracking of the particle.

Input File Usage: Use the following option to specify a fixed size for the particle tracking box in a DEM analysis:

**SECTION CONTROLS*
blank line
blank line
X, Y, and Z-coordinates (lower box corner) and X, Y, and Z-coordinates (upper box corner), 1

Use the following option to specify a fixed size for the particle tracking box in an SPH analysis:

**SECTION CONTROLS*
first data line
second data line
X, Y, and Z-coordinates (lower box corner) and X, Y, and Z-coordinates (upper box corner), 1

Abaqus/CAE Usage: In Abaqus/CAE you can only specify section controls for SPH parameters in Abaqus/Explicit analyses involving the conversion of continuum elements to SPH particles.

Using Section Controls for Smoothed Particle Hydrodynamics (SPH)

In addition to controlling the size of the particle tracking box, you can control other aspects of the smoothed particle hydrodynamic (SPH) formulation implemented in Abaqus/Explicit.

Using Section Controls for Specifying the SPH Kernel

For a smoothed particle hydrodynamic analysis, you can choose the order of the kernel used for interpolation. For a list of references that discuss the various kernels that can be used, see *Smoothed Particle Hydrodynamics*.

Input File Usage: Use one of the following options:

**SECTION CONTROLS*, KERNEL=CUBIC
**SECTION CONTROLS*, KERNEL=QUADRATIC
**SECTION CONTROLS*, KERNEL=QUINTIC

Abaqus/CAE Usage: In Abaqus/CAE you can choose the order of the kernel used for interpolation only in Abaqus/Explicit analyses involving the conversion of continuum elements to SPH particles.

Mesh module: **Mesh->Element Type**: **Conversion to particles**: **Kernel**: **Cubic**, **Quadratic**, or **Quintic**

Using Section Controls for Specifying the SPH Formulation

By default, the SPH kernels satisfy the zero-order completeness requirement. A first-order complete corrected (normalized) kernel is also available, which is sometimes referred in the literature as the normalized SPH (NSPH) method. In high-deformation solid mechanics analyses the use of the NSPH method might lead to more accurate results.

In the SPH methods, a mean velocity filtering coefficient can be used for the modified coordinate updates for particles. A nonzero value for this coefficient leads to the XSPH method, as discussed in *Smoothed Particle Hydrodynamics*.

Input File Usage: Use one of the following options to specify the SPH formulation:

**SECTION CONTROLS*, SPH FORMULATION=CLASSICAL (default)
**SECTION CONTROLS*, SPH FORMULATION=NSPH
**SECTION CONTROLS*, SPH FORMULATION=XSPH

Abaqus/CAE Usage: In Abaqus/CAE you can only specify section controls for SPH parameters in Abaqus/Explicit analyses involving the conversion of continuum elements to SPH particles.

Using Section Controls for Specifying SPH Parameters

You can control the way the smoothing length is computed (see *Smoothed Particle Hydrodynamics*). You can specify the smoothing length (units of length) for precise control of the radius of influence associated with a given particle. Alternatively, you can scale the default smoothing length by specifying a dimensionless smoothing length factor. By default, the smoothing length is kept constant throughout the analysis. You can specify a variable smoothing length that increases or decreases during the analysis depending on the divergence of the velocity field, which is a measure of compressive or expansive behavior.

You can also specify the minimum number of particles within the sphere of influence for the given particle. If the total number of particles within the sphere of influence for the given particle is less than the specified minimum number of particles, the deformation gradient for this given particle is frozen, that is, unchanged between the previous and current time increment. In solid mechanics it means that the strain associated with this element will not be changed during the current time increment.

You can specify a mean velocity filtering coefficient that is used for the modified coordinate updates for particles using the XSPH method.

Input File Usage: Use the following option to specify SPH parameters:

**SECTION CONTROLS*
first data line
smoothing length, *smoothing length factor*,
min number of neighboring particles, , *mean velocity filtering coefficient*

Use one of the following options to define the smoothing length:

**SECTION CONTROLS*, SPH SMOOTHING LENGTH=CONSTANT (default)
**SECTION CONTROLS*, SPH SMOOTHING LENGTH=VARIABLE

Abaqus/CAE Usage: In Abaqus/CAE you can only specify section controls for SPH parameters in Abaqus/Explicit analyses involving the conversion of continuum elements to SPH particles.

Using Section Controls to Convert Continuum Elements to Particles

Reduced-integration continuum elements can convert to particles if a certain criterion is met, as discussed in *Finite Element Conversion to SPH Particles*. You can specify the number of particles per parent element to be generated. Several criteria to trigger the conversion are available.

Input File Usage: Use the following option to prevent finite elements from converting to particles:

**SECTION CONTROLS*, ELEMENT CONVERSION=NO (default)

Use the following option to trigger the conversion of finite elements to particles based on a uniform background grid:

**SECTION CONTROLS*, ELEMENT CONVERSION=YES, SPH CONVERSION=BACKGROUND GRID (default)

Use the following option to trigger the conversion of finite elements to particles based on per element conversion:

**SECTION CONTROLS*, ELEMENT CONVERSION=YES, SPH CONVERSION=PER ELEMENT

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Conversion to particles**: **Background grid** or **Per element**

Specifying the Number of Particles Generated

You specify the number of particles to be generated per isoparametric direction. The number of particles can range from 1 to 7.

Input File Usage: **SECTION CONTROLS*, ELEMENT CONVERSION=YES, SPH CONVERSION=PER ELEMENT
first data line
second data line
third data line
number of particles to be generated per isoparametric direction

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Conversion to particles**: **Per element**, **PPD**: *number of particles to be generated per isoparametric direction*

Specifying the Background Grid

You specify the spacing of the background grid and the name of an orientation definition to define a local coordinate system for the background grid.

Input File Usage: **SECTION CONTROLS*, ELEMENT CONVERSION=YES, SPH CONVERSION=BACKGROUND GRID (default)
first data line
second data line
third data line
spacing of the background grid, name of an orientation definition

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Conversion to particles**: **Background grid**, **Spacing**

Specifying an orientation to define a local coordinate system for the background grid is not supported in Abaqus/CAE.

Specifying the Thickness of Generated Particles

The thickness of the particles is primarily used in resolving initial overclosures between the particles and the surfaces in the general contact. When particles are generated based on the uniform background method, you can specify the thickness of the generated particles to be either variable or uniform.

Input File Usage: Use one of the following options to define the thickness of the generated particles:

**SECTION CONTROLS*, PARTICLE THICKNESS=VARIABLE (default)
**SECTION CONTROLS*, PARTICLE THICKNESS=UNIFORM

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Conversion to particles**: **Background grid**, **Thickness**

Specifying a Time-Based Criterion

The time-based criterion is primarily intended as a modeling tool to allow all particles to convert from the defined finite element mesh at the same time.

Input File Usage:
**SECTION CONTROLS*, ELEMENT CONVERSION=YES, SPH
CONVERSION=PER ELEMENT
CONVERSION CRITERION=TIME (default)
first data line
second data line
third data line
, *time of conversion*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Conversion to particles**: **Per element**, **Criterion**: **Time**

Specifying a Strain-Based Criterion

The strain-based criterion is primarily intended for cases in which you want to use a progressive conversion approach. You specify the maximum principle strain (absolute value) when continuum elements are to convert to SPH particles.

Input File Usage:
**SECTION CONTROLS*, ELEMENT CONVERSION=YES, SPH
CONVERSION=PER ELEMENT
CONVERSION CRITERION=STRAIN
first data line
second data line
third data line
, *maximum principle strain (absolute value)*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Conversion to particles**: **Per element**, **Criterion**: **Strain**

Specifying a Stress-Based Criterion

Similar to the strain-based criterion, the stress-based criterion is primarily intended for cases in which you want to use a progressive conversion approach. You specify the maximum principle stress (absolute value) when continuum elements are to convert to SPH particles.

Input File Usage: **SECTION CONTROLS*, ELEMENT CONVERSION=YES, SPH CONVERSION=PER ELEMENT
CONVERSION CRITERION=STRESS
first data line
second data line
third data line
, *maximum principle stress (absolute value)*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Conversion to particles**: **Per element**, **Criterion**: **Stress**

Specifying a User Subroutine–Based Criterion

The user subroutine–based criterion allows you to implement a user-defined conversion criterion. You can control element conversion during the course of an Abaqus/Explicit analysis through any of the user subroutines that can actively modify state variables associated with a material point, such as *VUSDFLD* and *VUMAT*.

Input File Usage: Use the following option to trigger a user subroutine–based conversion criterion:

**SECTION CONTROLS*, CONVERSION CRITERION=USER, ELEMENT CONVERSION=YES, SPH CONVERSION=PER ELEMENT

Abaqus/CAE Usage: Specifying a user subroutine–based criterion for element conversion is not supported in Abaqus/CAE.

Using Section Controls for Bending Stabilization

Full-integration linear elements using hyperelastic materials might develop spurious hourglass-like deformation patterns when subjected to bending. The Abaqus formulation for these elements includes a stabilization algorithm that helps prevent the spurious modes. The stabilization is based on the initial shear modulus of the material and provides an additional scaling factor that might optionally be utilized to adjust the amount of stabilization for a given material.

In Abaqus/Standard bending stabilization is available for the following element types:

- CPE4
- CPE4H
- CPEG4
- CPEG4H
- CAX4
- CGAX4
- CAX4H
- CGAX4H
- C3D8
- C3D8S
- C3D8H

- C3D8HS
- QEC3D8

By default, the stabilization is active when the material response is modeled using hyperelastic (without hysteresis), hyperfoam, or low-density foam material models. Optionally, you can activate it for user-defined materials that model a hyperelastic response utilizing user subroutines *UHYPER*, *UHYPER_STRETCH*, or *UMAT*. Bending stabilization is also supported for materials modeled using the multiscale approach, as long as one or more of the constituents are defined using hyperelastic (without hysteresis), hyperfoam, or low-density foam material models.

In Abaqus/Explicit bending stabilization is relevant for C3D8 elements only. By default, it is active for C3D8 elements modeled with built-in hyperelastic, anisotropic hyperelastic (built-in and user-defined models utilizing user subroutines *VUANISOHYPER_INV* or *VUANISOHYPER_STRAIN*), and hyperfoam materials. Bending stabilization is inactive by default for user-defined materials defined with user subroutine *VUMAT*. You can use section controls to turn bending stabilization on or off and to specify an optional linear scaling factor to adjust the level of stabilization.

Input File Usage: Use the following option to activate bending stabilization:

**SECTION CONTROLS*, NAME=*name*, BENDING STABILIZATION=ON

Use the following option to deactivate bending stabilization:

**SECTION CONTROLS*, NAME=*name*, BENDING STABILIZATION=OFF

Use the following options to specify a scale factor (f_s) for the bending stabilization:

**SECTION CONTROLS*, NAME=*name*, BENDING STABILIZATION=ON
, , , , , ,*scale factor for bending stabilization*

Continuum Elements

In this section:

- *General-Purpose Continuum Elements*
- *Infinite Elements*
- *Warping Elements*

General-Purpose Continuum Elements

In this section:

- *Solid (Continuum) Elements*
- *One-Dimensional Solid (Link) Element Library*
- *Two-Dimensional Solid Element Library*
- *Three-Dimensional Solid Element Library*
- *Cylindrical Solid Element Library*
- *Axisymmetric Solid Element Library*
- *Axisymmetric Solid Elements with Nonlinear, Asymmetric Deformation*

Solid (Continuum) Elements

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Choosing the Element's Dimensionality*
- *One-Dimensional Solid (Link) Element Library*
- *Two-Dimensional Solid Element Library*
- *Three-Dimensional Solid Element Library*
- *Cylindrical Solid Element Library*
- *Axisymmetric Solid Element Library*
- *Axisymmetric Solid Elements with Nonlinear, Asymmetric Deformation*
- **SOLID SECTION*
- **HOURGLASS STIFFNESS*
- *Creating homogeneous solid sections*
- *Creating composite solid sections*
- *Creating electromagnetic solid sections*
- *Assigning a material orientation*
- *Composite layups*

Overview

Solid (continuum) elements:

- are the standard volume elements of Abaqus;
- do not include structural elements such as beams, shells, membranes, and trusses; special-purpose elements such as gap elements; or connector elements such as connectors, springs, and dashpots;
- can be composed of a single homogeneous material or, in Abaqus/Standard, can include several layers of different materials for the analysis of laminated composite solids; and
- are more accurate if not distorted, particularly for quadrilaterals and hexahedra. The triangular and tetrahedral elements are less sensitive to distortion.

Typical Applications

The solid (or continuum) elements in Abaqus can be used for linear analysis and for complex nonlinear analyses involving contact, plasticity, and large deformations. They are available for stress, heat transfer, acoustic, coupled thermal-stress, coupled pore fluid-stress, piezoelectric, magnetostatic, electromagnetic, and coupled thermal-electrical analyses (see *Choosing the Appropriate Element for an Analysis Type*).

Choosing an Appropriate Element

There are some differences in the solid element libraries available in Abaqus/Standard and Abaqus/Explicit.

Abaqus/Standard solid element library

The Abaqus/Standard solid element library includes first-order (linear) interpolation elements and second-order (quadratic) interpolation elements in one, two, or three dimensions. Triangles and quadrilaterals

are available in two dimensions. Tetrahedra, wedges, and hexahedra (“bricks”) are provided in three dimensions. Modified second-order triangular and tetrahedral elements are also provided.

Curved (parabolic) edges can be used on the quadratic elements but are not recommended for pore pressure or coupled temperature-displacement elements. Cylindrical elements are provided for structures with edges that are initially circular.

In addition, reduced-integration, hybrid, and incompatible mode elements are available in Abaqus/Standard.

Electromagnetic elements, based on an edge-based interpolation of the magnetic vector potential, are provided both in two and three dimensions.

Abaqus/Explicit solid element library

The Abaqus/Explicit solid element library includes first-order (linear) interpolation elements and modified second-order interpolation elements in two or three dimensions. In addition, the element library also includes a second-order interpolation tetrahedral element (C3D10) in three dimensions. Triangular and quadrilateral first-order elements are available in two dimensions. Tetrahedral, wedge, and hexahedral (“brick”) first-order elements are available in three dimensions. The modified second-order elements are limited to triangles and tetrahedra. The acoustic elements in Abaqus/Explicit are limited to first-order (linear) interpolations. For incompatible mode elements only three-dimensional elements are available.

The second-order tetrahedral element C3D10 in Abaqus/Explicit is subject to the following limitations:

- It does not support selective subcycling.
- If any midedge node of a C3D10 element is not on the center of the edge, the location of the midedge node is adjusted.

Various two-dimensional elements (plane stress, plane strain, axisymmetric) are available in both Abaqus/Standard and Abaqus/Explicit. See *Choosing the Element's Dimensionality* for details.

Given the wide variety of element types available, it is important to select the correct element for a particular application. Choosing an element for a particular analysis can be simplified by considering specific element characteristics: first- or second-order; full or reduced integration; hexahedra/quadrilaterals or tetrahedra/triangles; or normal, hybrid, or incompatible mode formulation. By considering each of these aspects carefully, the best element for a given analysis can be selected.

Choosing between First- and Second-Order Elements

In first-order plane strain, generalized plane strain, axisymmetric quadrilateral, hexahedral solid elements, and cylindrical elements, the strain operator provides constant volumetric strain throughout the element. This constant strain prevents mesh “locking” when the material response is approximately incompressible (see *Solid isoparametric quadrilaterals and hexahedra* for a more detailed discussion).

Second-order elements provide higher accuracy than first-order elements for “smooth” problems that do not involve severe element distortions. They capture stress concentrations more effectively and are better for modeling geometric features: they can model a curved surface with fewer elements. Finally, second-order elements are very effective in bending-dominated problems.

First-order triangular and tetrahedral elements should be avoided as much as possible in stress analysis problems; the elements are overly stiff and exhibit slow convergence with mesh refinement, which is especially a problem with first-order tetrahedral elements. If they are required, an extremely fine mesh might be needed to obtain results of sufficient accuracy.

Choosing between Full- and Reduced-Integration Elements

Reduced integration uses a lower-order integration to form the element stiffness. The mass matrix and distributed loadings use full integration. Reduced integration reduces running time, especially in three dimensions. For

example, element type C3D20 has 27 integration points, while C3D20R has only 8; therefore, element assembly is roughly 3.5 times more costly for C3D20 than for C3D20R.

In Abaqus/Standard you can choose between full or reduced integration for quadrilateral and hexahedral (brick) elements. In Abaqus/Explicit you can choose between full or reduced integration for hexahedral (brick) elements. Only reduced-integration first-order elements are available for quadrilateral elements in Abaqus/Explicit; the elements with reduced integration are also referred to as uniform strain or centroid strain elements with hourglass control.

Second-order reduced-integration elements in Abaqus/Standard generally yield more accurate results than the corresponding fully integrated elements. However, for first-order elements the accuracy achieved with full versus reduced integration is largely dependent on the nature of the problem.

Hourglassing

Hourglassing can be a problem with first-order, reduced-integration elements (CPS4R, CAX4R, C3D8R, etc.) in stress/displacement analyses. Since the elements have only one integration point, it is possible for them to distort in such a way that the strains calculated at the integration point are all zero, which, in turn, leads to uncontrolled distortion of the mesh. First-order, reduced-integration elements in Abaqus include hourglass control, but they should be used with reasonably fine meshes. Hourglassing can also be minimized by distributing point loads and boundary conditions over a number of adjacent nodes.

In Abaqus/Standard the second-order reduced-integration elements, with the exception of the 27-node C3D27R and C3D27RH elements, do not have the same difficulty and are recommended in all cases when the solution is expected to be smooth. The C3D27R and C3D27RH elements have three unconstrained, propagating hourglass modes when all 27 nodes are present. These elements should not be used with all 27 nodes, unless they are sufficiently constrained through boundary conditions. First-order elements are recommended when large strains or very high strain gradients are expected.

Shear and Volumetric Locking

Fully integrated elements in Abaqus/Standard and Abaqus/Explicit do not hourglass but might suffer from "locking" behavior: both shear and volumetric locking. Shear locking occurs in first-order, fully integrated elements (CPS4, CPE4, C3D8, etc.) that are subjected to bending. The numerical formulation of the elements gives rise to shear strains that do not really exist—the so-called parasitic shear. Therefore, these elements are too stiff in bending, in particular if the element length is of the same order of magnitude as or greater than the wall thickness. See *Performance of continuum and shell elements for linear analysis of bending problems* for further discussion of the bending behavior of solid elements.

Volumetric locking occurs in fully integrated elements when the material behavior is (almost) incompressible. Spurious pressure stresses develop at the integration points, causing an element to behave too stiffly for deformations that should cause no volume changes. If materials are almost incompressible (elastic-plastic materials for which the plastic strains are incompressible), second-order, fully integrated elements start to develop volumetric locking when the plastic strains are on the order of the elastic strains. However, the first-order, fully integrated quadrilaterals and hexahedra use selectively reduced integration (reduced integration on the volumetric terms). Therefore, these elements do not lock with almost incompressible materials. Reduced-integration, second-order elements develop volumetric locking for almost incompressible materials only after significant straining occurs. In this case, volumetric locking is often accompanied by a mode that looks like hourglassing. Frequently, this problem can be avoided by refining the mesh in regions of large plastic strain.

If volumetric locking is suspected, check the pressure stress at the integration points (printed output). If the pressure values show a checkerboard pattern, changing significantly from one integration point to the next, volumetric locking is occurring. Choosing a quilt-style contour plot in the Visualization module of Abaqus/CAE will show the effect.

Specifying Nondefault Section Controls

You can specify a nondefault hourglass control formulation or scale factor for reduced-integration first-order elements (4-node quadrilaterals and 8-node bricks with one integration point). See *Section Controls* for more information about section controls.

In Abaqus/Explicit section controls can also be used to specify a nondefault kinematic formulation for 8-node brick elements, the accuracy order of the element formulation, and distortion control for either 4-node quadrilateral or 8-node brick elements. Section controls are also used with coupled temperature-displacement elements in Abaqus/Explicit to change the default values for the mechanical response analysis.

In Abaqus/Standard you can specify nondefault hourglass stiffness factors based on the default total stiffness approach for reduced-integration first-order elements (4-node quadrilaterals and 8-node bricks with one integration point) and modified tetrahedral and triangular elements.

There are no hourglass stiffness factors or scale factors for the nondefault enhanced hourglass control formulation. See *Section Controls* for more information about hourglass control.

Input File Usage: Use both of the following options to associate a section control definition with the element section definition:

**SECTION CONTROLS*, NAME=*name*
**SOLID SECTION*, CONTROLS=*name*

Use both of the following options in Abaqus/Standard to specify nondefault hourglass stiffness factors for the total stiffness approach:

**SOLID SECTION*
**HOURGLASS STIFFNESS*

Abaqus/CAE Usage: Mesh module:
Element Type: **Element Controls**
Element Type: **Hourglass stiffness**: **Specify**

Choosing between Bricks/Quadrilaterals and Tetrahedra/Triangles

Triangular and tetrahedral elements are geometrically versatile and are used in many automatic meshing algorithms. It is very convenient to mesh a complex shape with triangles or tetrahedra, and the second-order and modified triangular and tetrahedral elements (CPE6, CPE6M, C3D10, C3D10M, etc.) in Abaqus are suitable for general usage. However, a good mesh of hexahedral elements usually provides a solution of equivalent accuracy at less cost. Quadrilaterals and hexahedra have a better convergence rate than triangles and tetrahedra, and sensitivity to mesh orientation in regular meshes is not an issue. However, triangles and tetrahedra are less sensitive to initial element shape, whereas first-order quadrilaterals and hexahedra perform better if their shape is approximately rectangular. The elements become much less accurate when they are initially distorted (see *Performance of continuum and shell elements for linear analysis of bending problems*).

First-order triangles and tetrahedra are usually overly stiff, and extremely fine meshes are required to obtain accurate results. As mentioned earlier, fully integrated first-order triangles and tetrahedra in Abaqus/Standard also exhibit volumetric locking in incompressible problems. As a rule, these elements should not be used except as filler elements in noncritical areas. Therefore, try to use well-shaped elements in regions of interest.

Tetrahedral and Wedge Elements

For stress/displacement analyses the first-order tetrahedral element C3D4 is a constant stress tetrahedron, which should be avoided as much as possible; the element exhibits slow convergence with mesh refinement. This element provides accurate results only in general cases with very fine meshing. Therefore, C3D4 is recommended

only for filling in regions of low stress gradient in meshes of C3D8 or C3D8R elements, when the geometry precludes the use of C3D8 or C3D8R elements throughout the model. For tetrahedral element meshes the second-order or the modified tetrahedral elements, C3D10 or C3D10M, should be used. In Abaqus/Explicit the second-order tetrahedral element C3D10 has better accuracy and performance than the modified tetrahedral element C3D10M.

Similarly, the linear version of the wedge element C3D6 should generally be used only when necessary to complete a mesh, and, even then, the element should be far from any areas where accurate results are needed. This element provides accurate results only with very fine meshing.

Modified Triangular and Tetrahedral Elements

A family of modified 6-node triangular and 10-node tetrahedral elements is available that provides improved performance over the first-order triangular and tetrahedral elements and that occasionally provides improved behavior to regular second-order triangular and tetrahedral elements. Regular second-order triangular and tetrahedral elements are typically preferable in Abaqus/Standard; however, regular second-order triangular and tetrahedral elements might exhibit "volumetric locking" when incompressibility is approached, such as in problems with a large amount of plastic deformation. As discussed in *Three-Dimensional Surfaces with Second-Order Faces and a Node-to-Surface Formulation*, regular second-order tetrahedral elements cannot underly a secondary surface for the node-to-surface contact formulation with strict enforcement of a "hard" contact relationship. This limitation is typically not significant because the surface-to-surface contact formulation and penalty contact enforcement are generally recommended.

Modified triangular and tetrahedral elements work well in contact, exhibit minimal shear and volumetric locking, and are robust during finite deformation (see *The Hertz contact problem* and *Upsetting of a cylindrical billet: coupled temperature-displacement and adiabatic analysis*). These elements use a lumped matrix formulation for dynamic analysis. Modified triangular elements are provided for planar and axisymmetric analysis, and modified tetrahedra are provided for three-dimensional analysis. In addition, hybrid versions of these elements are provided in Abaqus/Standard for use with incompressible and nearly incompressible constitutive models.

When the total stiffness approach is chosen, modified tetrahedral and triangular elements (C3D10M, CPS6M, CAX6M, etc.) use hourglass control associated with their internal degrees of freedom. The hourglass modes in these elements do not usually propagate; hence, the hourglass stiffness is usually not as significant as for first-order elements.

For most Abaqus/Standard analysis models the same mesh density appropriate for the regular second-order triangular and tetrahedral elements can be used with the modified elements to achieve similar accuracy. For comparative results, see the following:

- *Geometrically nonlinear analysis of a cantilever beam*
- *Performance of continuum and shell elements for linear analysis of bending problems*
- *LE1: Plane stress elements—elliptic membrane*
- *LE10: Thick plate under pressure*
- *FV32: Cantilevered tapered membrane*
- *FV52: Simply supported "solid" square plate*

However, in analyses involving thin bending situations with finite deformations (see *Pressurized rubber disc*) and in frequency analyses where high bending modes need to be captured accurately (see *FV41: Free cylinder: axisymmetric vibration*), the mesh has to be more refined for the modified triangular and tetrahedral elements (by at least one and a half times) to attain accuracy comparable to the regular second-order elements.

The modified triangular and tetrahedral elements might not be adequate to be used in the coupled pore fluid diffusion and stress analysis in the presence of large pore pressure fields if enhanced hourglass control is used.

The modified elements are more expensive computationally than lower-order quadrilaterals and hexahedron and sometimes require a more refined mesh for the same level of accuracy. However, in Abaqus/Explicit they are provided as an attractive alternative to the lower-order triangles and tetrahedron to take advantage of automatic triangular and tetrahedral mesh generators.

Compatibility with Other Elements

The modified triangular and tetrahedral elements are incompatible with the regular second-order solid elements in Abaqus/Standard. Thus, they should not be connected with these elements in a mesh.

Surface Stress Output

In areas of high stress gradients, stresses extrapolated from the integration points to the nodes are not as accurate for the modified elements as for similar second-order triangles and tetrahedra in Abaqus/Standard. In cases where more accurate surface stresses are needed, the surface can be coated with membrane elements that have a significantly lower stiffness than the underlying material. The stresses in these membrane elements will then reflect more accurately the surface stress and can be used for output purposes.

Fully Constrained Displacements

In Abaqus/Standard if all the displacement degrees of freedom on all the nodes of a modified element are constrained with boundary conditions, a similar boundary condition is applied to an internal node in the element. If a distributed load is subsequently applied to this element, the reported reaction forces at the nodes you defined will not sum up to the applied load since some of the applied load is taken by the internal node whose reaction force is not reported.

Choosing between Regular and Hybrid Elements

Hybrid elements are intended primarily for use with incompressible and almost incompressible material behavior; these elements are available only in Abaqus/Standard. When the material response is incompressible, the solution to a problem cannot be obtained in terms of the displacement history only, since a purely hydrostatic pressure can be added without changing the displacements.

Almost Incompressible Material Behavior

Near-incompressible behavior occurs when the bulk modulus is very much larger than the shear modulus (for example, in linear elastic materials where the Poisson's ratio is greater than .48) and exhibits behavior approaching the incompressible limit: a very small change in displacement produces extremely large changes in pressure. Therefore, a purely displacement-based solution is too sensitive to be useful numerically (for example, computer round-off might cause the method to fail).

This singular behavior is removed from the system by treating the pressure stress as an independently interpolated basic solution variable, coupled to the displacement solution through the constitutive theory and the compatibility condition. This independent interpolation of pressure stress is the basis of the hybrid elements. Hybrid elements have more internal variables than their nonhybrid counterparts and are slightly more expensive. See *Hybrid incompressible solid element formulation* for further details.

Fully Incompressible Material Behavior

Hybrid elements must be used if the material is fully incompressible (except in the case of plane stress since the incompressibility constraint can be satisfied by adjusting the thickness). If the material is almost incompressible and hyperelastic, hybrid elements are still recommended. For almost incompressible, elastic-plastic materials and for compressible materials, hybrid elements offer insufficient advantage and, hence, should not be used.

For Mises and Hill plasticity the plastic deformation is fully incompressible; therefore, the rate of total deformation becomes incompressible as the plastic deformation starts to dominate the response. All of the quadrilateral and

brick elements in Abaqus/Standard can handle this rate-incompressibility condition except for the fully integrated quadrilateral and brick elements without the hybrid formulation: CPE8, CPEG8, CAX8, CGAX8, and C3D20. These elements will “lock” (become overconstrained) as the material becomes more incompressible.

Elastic Strains in Hybrid Elements

Hybrid elements use an independent interpolation for the hydrostatic pressure, and the elastic volumetric strain is calculated from the pressure. Hence, the elastic strains agree exactly with the stress, but they agree with the total strain only in an element average sense and not pointwise, even if no inelastic strains are present. For isotropic materials this behavior is noticeable only in second-order, fully integrated hybrid elements. In these elements the hydrostatic pressure (and, thus, the volumetric strain) varies linearly over the element, whereas the total strain might exhibit a quadratic variation.

For anisotropic materials this behavior also occurs in first-order, fully integrated hybrid elements. In such materials there is typically a strong coupling between volumetric and deviatoric behavior: volumetric strain will give rise to deviatoric stresses and, conversely, deviatoric strains will give rise to hydrostatic pressure. Hence, the constant hydrostatic pressure enforced in the fully integrated, first-order hybrid elements does not generally yield a constant elastic strain; whereas the total volume strain is always constant for these elements, as discussed earlier in this section. Therefore, hybrid elements are not recommended for use with anisotropic materials unless the material is approximately incompressible, which usually implies that the coupling between deviatoric and volume behavior is relatively weak.

Using Hybrid Elements with Material Models That Exhibit Volumetric Plasticity

If the material model exhibits volumetric plasticity, such as the (capped) Drucker-Prager model, slow convergence or convergence problems might occur if second-order hybrid elements are used. In that case good results can usually be obtained with regular (nonhybrid) second-order elements.

Determining the Need for Hybrid Elements

For nearly incompressible materials a displaced shape plot that shows a more or less homogeneous but nonphysical pattern of deformation is an indication of mesh locking. As previously discussed, fully integrated elements should be changed to reduced-integration elements in this case. If reduced-integration elements are already being used, the mesh density should be increased. Finally, hybrid elements can be used if problems persist.

Hybrid Triangular and Tetrahedral Elements

The following hybrid, triangular, two-dimensional, and axisymmetric elements should be used only for mesh refinement or to fill in regions of meshes of quadrilateral elements: CPE3H, CPEG3H, CAX3H, and CGAX3H. Hybrid, three-dimensional tetrahedral elements C3D4H and wedge elements C3D6H should be used only for mesh refinement or to fill in regions of meshes of brick-type elements. Since each C3D6H element introduces a constraint equation in a fully incompressible problem, a mesh containing only these elements will be overconstrained. Abutting regions of C3D4H elements with different material properties should be tied rather than sharing nodes to allow discontinuity jumps in the pressure and volumetric fields.

In addition, the second-order three-dimensional hybrid elements C3D10H, C3D10MH, C3D15H, and C3D15VH are significantly more expensive than their nonhybrid counterparts.

Multi-Purpose, Improved Surface Stress Visualization Tetrahedra

The C3D10HS tetrahedron has been developed for improved bending results in coarse meshes while avoiding pressure locking in metal plasticity and quasi-incompressible and incompressible rubber elasticity. These elements are available only in Abaqus/Standard. Internal pressure degrees of freedom are activated automatically for a given element once the material exhibits behavior approaching the incompressible limit (that is, an effective Poisson's ratio above .45). This unique feature of C3D10HS elements make it especially suitable for modeling metal plasticity, since it activates the pressure degrees of freedom only in the regions of the model where the

material is incompressible. With activation of the internal degrees of freedom, C3D10HS elements have more variables than either hybrid (C3D10H) or nonhybrid (C3D10) elements; therefore, they are more expensive. This element also uses a unique 11-point integration scheme, providing a superior stress visualization scheme in coarse meshes because it avoids errors due to the extrapolation of stress components from the integration points to the nodes.

Improved Surface Stress Visualization Bricks

The C3D8S and C3D8HS linear brick elements have been developed to provide a superior stress visualization on the element surface by avoiding errors due to the extrapolation of stress components from the integration points to the nodes. These elements are available only in Abaqus/Standard. The C3D8S and C3D8HS elements have the same degrees of freedom and use the same element linear interpolation as C3D8 and C3D8H, respectively. These elements use a 27-point integration scheme consisting of 8 integration points at the elements' nodes, 12 integration points on the elements' edges, 6 integration points on the elements' sides, and one integration point inside the element. To reduce the size of the output database, you can request element output at the nodes. Because these elements have integration points at the nodes, there is no error associated with extrapolating integration point output variables to the nodes.

Incompatible Mode Elements

Incompatible mode elements (CPS4I, CPE4I, CAX4I, CPEG4I, and C3D8I and the corresponding hybrid elements) are first-order elements that are enhanced by incompatible modes to improve their bending behavior; all of these elements are available in Abaqus/Standard and only element C3D8I is available in Abaqus/Explicit.

In addition to the standard displacement degrees of freedom, incompatible deformation modes are added internally to the elements. The primary effect of these modes is to eliminate the parasitic shear stresses that cause the response of the regular first-order displacement elements to be too stiff in bending. In addition, these modes eliminate the artificial stiffening due to Poisson's effect in bending (which is manifested in regular displacement elements by a linear variation of the stress perpendicular to the bending direction). In the nonhybrid elements—except for the plane stress element, CPS4I—additional incompatible modes are added to prevent locking of the elements with approximately incompressible material behavior. For fully incompressible material behavior the corresponding hybrid elements must be used.

Because of the added internal degrees of freedom due to the incompatible modes (4 for CPS4I; 5 for CPE4I, CAX4I, and CPEG4I; and 13 for C3D8I), these elements are somewhat more expensive than the regular first-order displacement elements; however, they are significantly more economical than second-order elements. The incompatible mode elements use full integration and, thus, have no hourglass modes.

Incompatible mode elements are discussed in more detail in *Continuum elements with incompatible modes*.

Shape Considerations

The incompatible mode elements perform almost as well as second-order elements in many situations if the elements have an approximately rectangular shape. The performance is reduced considerably if the elements have a parallelogram shape. The performance of trapezoidal-shaped incompatible mode elements is not much better than the performance of the regular, fully integrated, first-order interpolation elements; see *Performance of continuum and shell elements for linear analysis of bending problems*, which illustrates the loss of accuracy associated with distorted elements.

Using Incompatible Mode Elements in Large-Strain Applications

Incompatible mode elements should be used with caution in applications involving large compressive strains. Convergence might be slow at times, and inaccuracies might accumulate in hyperelastic applications. Hence, erroneous residual stresses might sometimes appear in hyperelastic elements that are unloaded after having been subjected to a complex deformation history.

Using Incompatible Mode Elements with Regular Elements

Incompatible mode elements can be used in the same mesh with regular solid elements. Generally the incompatible mode elements should be used in regions where bending response must be modeled accurately, and they should be of rectangular shape to provide the most accuracy. While these elements often provide accurate response in such cases, it is generally preferable to use structural elements (shells or beams) to model structural components.

Continuum Solid Shell Elements

Continuum solid shell elements (CSS8) are first-order elements with seven incompatible modes to improve bending behavior and an assumed strain to mitigate locking. The formulation is based on work by *Vu-Quoc and Tan (2003)*. The CSS8 elements are well suited for thin structural applications, including composites, where you want a full three-dimensional constitutive response. They fill a gap between incompatible mode elements, which use three-dimensional constitutive laws but tend to lock in bending for large aspect ratios, and continuum shell elements, which have a very good bending response for large aspect ratios but are restricted to two-dimensional plane stress constitutive behaviors. The continuum solid shell elements are available only in Abaqus/Standard.

Continuum solid shell elements have only displacement degrees of freedom and are fully compatible with regular continuum elements. They use full integration and have no hourglass modes. The assumed strain leads to a through-thickness response that is different from the in-plane response; therefore, while these elements pass in-plane membrane and out-of-plane bending patch tests, they do not pass the standard three-dimensional patch test.

When a user-defined orientation is used with continuum solid shell elements, the orientation is projected onto the element midsurface in the initial configuration, such that the local material 3-direction is normal to the element midsurface. In geometrically nonlinear analyses the local directions will rotate with the average material rotation at each material point, the same as for other solid elements. If the element undergoes significant transverse shear deformation, the local 3-direction will no longer remain normal to the element midsurface.

In applications involving large strains, you should use continuum solid shell elements with caution. Convergence might be slow at times, and inaccuracies might accumulate in hyperelastic applications.

Normal Definition for Continuum Solid Shell Elements

Figure 1 illustrates a key modeling feature of continuum solid shell elements. Since the behavior in the thickness direction is different from that in the in-plane directions, it is important that the continuum solid shell elements are oriented properly. The element top and bottom faces (and, hence, the element normal, stacking direction, and thickness) are defined by the nodal connectivity. For continuum solid shell element CSS8 the face with corner nodes 1, 2, 3, and 4 is the bottom face, and the face with corner nodes 5, 6, 7, and 8 is the top face. The stacking direction and thickness direction are both defined to be the direction from the bottom to the top face. The thickness direction and the stacking direction should always be in the third direction of the element's isoparametric directions.

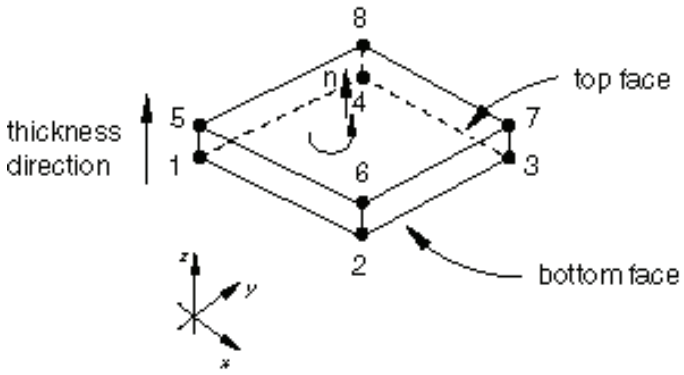


Figure 1: Normal and thickness direction for continuum solid shell elements.

Variable Node Elements

Variable node elements (such as C3D27 and C3D15V) allow midface nodes to be introduced on any element face (on any rectangular face only for the wedge C3D15V). The choice is made by the nodes specified in the element definition. These elements are available only in Abaqus/Standard and can be used quite generally in any three-dimensional model. The C3D27 family of elements is frequently used as the ring of elements around a crack line.

Cylindrical Elements

Cylindrical elements (CCL9, CCL9H, CCL12, CCL12H, CCL18, CCL18H, CCL24, CCL24H, and CCL24RH) are available only in Abaqus/Standard for precise modeling of regions in a structure with circular geometry, such as a tire. The elements make use of trigonometric functions to interpolate displacements along the circumferential direction and use regular isoparametric interpolation in the radial or cross-sectional plane of the element. All the elements use three nodes along the circumferential direction and can span angles between 0 and 180°. Elements with both first-order and second-order interpolation in the cross-sectional plane are available.

The geometry of the element is defined by specifying nodal coordinates in a global Cartesian system. The default nodal output is also provided in a global Cartesian system. Output of stress, strain, and other material point output quantities are done, by default, in a fixed local cylindrical system where direction 1 is the radial direction, direction 2 is the axial direction, and direction 3 is the circumferential direction. This default system is computed from the reference configuration of the element. An alternative local system can be defined (see *Orientations*). In this case the output of stress, strain, and other material point quantities is done in the oriented system.

The cylindrical elements can be used in the same mesh with regular elements. In particular, regular solid elements can be connected directly to the nodes on the cross-sectional plane of cylindrical elements. For example, any face of a C3D8 element can share nodes with the cross-sectional faces (faces 1 and 2; see *Cylindrical Solid Element Library* for a description of the element faces) of a CCL12 element. Regular elements can also be connected along the circular edges of cylindrical elements by using a surface-based tie constraint (*Mesh Tie Constraints*) provided that the cylindrical elements do not span a large segment. However, such usage might result in spurious oscillations in the solution near the tied surfaces and should be avoided when an accurate solution in this region is required.

Compatible membrane elements (*Membrane Elements*) and surface elements with rebar (*Surface Elements*) are available for use with cylindrical solid elements.

All elements with first-order interpolation in the cross-sectional plane use full integration for the deviatoric terms and reduced integration for the volumetric terms and, thus, have no hourglass modes and do not lock with almost incompressible materials. The hybrid elements with first-order and second-order interpolation in the cross-sectional plane use an independent interpolation for hydrostatic pressure.

Summary of Recommendations for Element Usage

The following recommendations apply to both Abaqus/Standard and Abaqus/Explicit:

- Make all elements as "well shaped" as possible to improve convergence and accuracy.
- If an automatic tetrahedral mesh generator is used, use the second-order elements C3D10 (in Abaqus/Standard) or C3D10 and C3D10M (in Abaqus/Explicit). Use the modified tetrahedral element C3D10M in Abaqus/Standard in analyses with large amounts of plastic deformation.
- If possible, use hexahedral elements in three-dimensional analyses since they give the best results for the minimum cost.

Abaqus/Standard users should also consider the following recommendations:

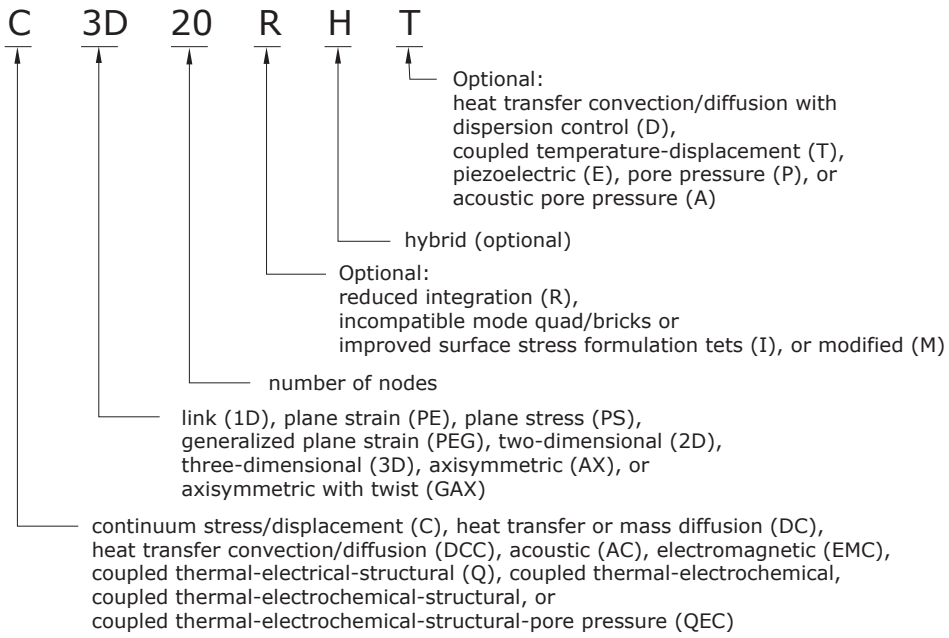
- For linear and "smooth" nonlinear problems use reduced-integration, second-order elements if possible.
- Use second-order, fully integrated elements close to stress concentrations to capture the severe gradients in these regions. However, avoid these elements in regions of finite strain if the material response is nearly incompressible.
- Use first-order quadrilateral or hexahedral elements or the modified triangular and tetrahedral elements for problems involving large distortions. If the mesh distortion is severe, use reduced-integration, first-order elements.
- If the problem involves bending and large distortions, use a fine mesh of first-order, reduced-integration elements.
- Hybrid elements must be used if the material is fully incompressible (except when using plane stress elements). Hybrid elements should also be used in some cases with nearly incompressible materials.
- Incompatible mode elements can give very accurate results in problems dominated by bending.

Naming Convention

The naming conventions for solid elements depend on the element dimensionality.

One-Dimensional, Two-Dimensional, Three-Dimensional, and Axisymmetric Elements

In general, one-dimensional, two-dimensional, three-dimensional, and axisymmetric solid elements in Abaqus are named as shown below. For example, CAX4R is an axisymmetric continuum stress/displacement, 4-node, reduced-integration element; and CPS8RE is an 8-node, reduced-integration, plane stress piezoelectric element.

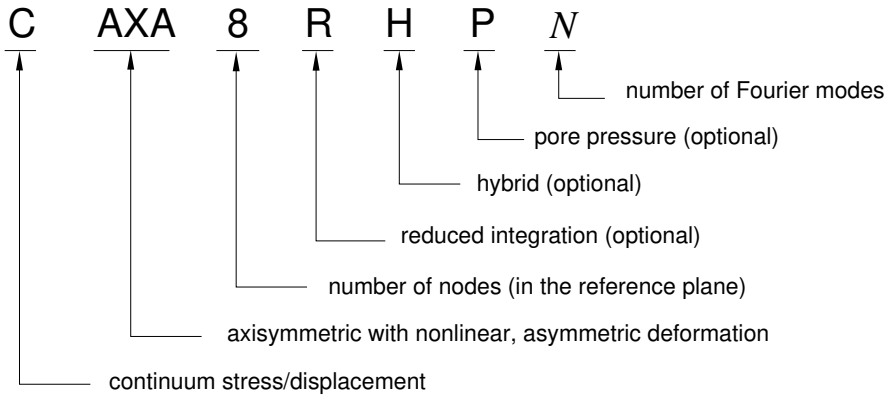


The following elements violate this naming convention:

- In Abaqus/Explicit, C3D6 and C3D6T elements, which are 6-node linear wedge, reduced integration elements.
- Continuum solid shell elements. For example, CSS8 is an 8-node linear brick, stress/displacement element with incompatible modes and assumed strain.
- Pore pressure elements (the hybrid elements have the letter H after the letter P). For example, CPE8PH is an 8-node, hybrid, plane strain, pore pressure element.

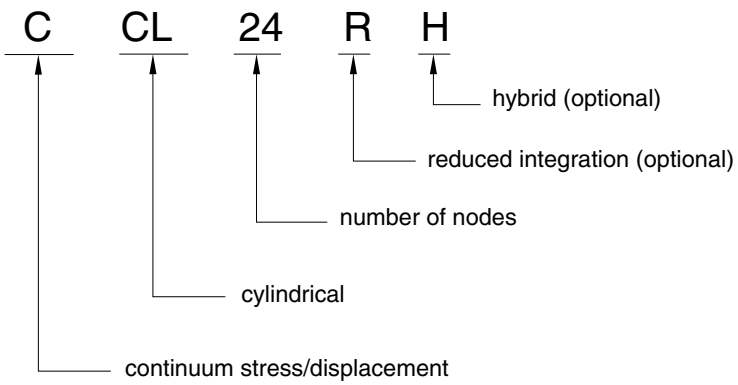
Axisymmetric Elements with Nonlinear Asymmetric Deformation

The axisymmetric solid elements with nonlinear asymmetric deformation in Abaqus/Standard are named as shown below. For example, CAXA4RH1 is a 4-node, reduced-integration, hybrid, axisymmetric element with nonlinear asymmetric deformation and one Fourier mode (see *Choosing the Element's Dimensionality*).



Cylindrical Elements

The cylindrical elements in Abaqus/Standard are named as shown below. For example, CCL24RH is a 24-node, hybrid, reduced-integration cylindrical element.



Defining the Elements Section Properties

A solid section definition is used to define the section properties of solid elements.

In Abaqus/Standard solid elements can be composed of a single homogeneous material or can include several layers of different materials for the analysis of laminated composite solids. In Abaqus/Explicit solid elements can be composed only of a single homogeneous material.

Defining Homogeneous Solid Elements

You must associate a material definition (*Material Data Definition*) with the solid section definition. In an Abaqus/Standard analysis spatially varying material behavior defined with one or more distributions (*Distribution Definition*) can be assigned to the solid section definition. In addition, you must associate the section definition with a region of your model.

In Abaqus/Standard if any of the material behaviors assigned to the solid section definition (through the material definition) are defined with distributions, spatially varying material properties are applied to all elements associated with the solid section. Default material behaviors (as defined by the distributions) are applied to any element that is not specifically included in the associated distribution.

Input File Usage: **SOLID SECTION*, MATERIAL=*name*, ELSET=*name*

where the ELSET parameter refers to a set of solid elements.

Abaqus/CAE Usage: Property module:
Create Section: select **Solid** as the section **Category** and **Homogeneous** or **Electromagnetic, Solid** as the section **Type**: **Material:** *name*
Assign->Section: select regions

Assigning an Orientation Definition

You can associate a material orientation definition with solid elements (see *Orientations*). A spatially varying local coordinate system defined with a distribution (*Distribution Definition*) can be assigned to the solid section definition.

If the orientation definition assigned to the solid section definition is defined with distributions, spatially varying local coordinate systems are applied to all elements associated with the solid section. A default local coordinate system (as defined by the distributions) is applied to any element that is not specifically included in the associated distribution.

Input File Usage: **SOLID SECTION*, ORIENTATION=*name*

Abaqus/CAE Usage: Property module: **Assign->Material Orientation**

Defining the Geometric Attributes, if Required

For some element types additional geometric attributes are required, such as the cross-sectional area for one-dimensional elements or the thickness for two-dimensional plane elements. The attributes required for a particular element type are defined in the solid element libraries. These attributes are given as part of the solid section definition.

Defining Composite Solid Elements in Abaqus/Standard

The use of composite solids is limited to three-dimensional brick elements, wedge elements, and the continuum solid shell elements that have only displacement degrees of freedom (they are not available for coupled temperature-displacement elements, piezoelectric elements, pore pressure elements, continuum cylindrical elements, and improved surface stress visualization elements). Composite solid elements are primarily intended for modeling convenience. Three-dimensional brick elements and wedge elements usually do not provide a more accurate solution than composite shell elements. However, the continuum solid shell elements are well suited for composite analyses and are comparable to composite shell elements.

The thickness, the number of section points required for numerical integration through each layer (discussed below), and the material name and orientation associated with each layer are specified as part of the composite solid section definition. In Abaqus/Standard you can use a distribution to specify spatially varying orientation angles and thicknesses on a layer (see *Distribution Definition*). A distribution that is used to define layer thickness must have a default value. The default layer thickness is used by any solid element assigned to the solid section that is not specifically assigned a value in the distribution.

For regular brick solid elements the material layers can be stacked in any of the three isoparametric coordinates, parallel to opposite faces of the isoparametric main element as shown in *Figure 2*. For continuum solid shell elements and wedge elements the material layers can be stacked only in the third isoparametric coordinates, parallel to opposite faces 1 and 2 of the isoparametric main element. The number of integration points within a layer at any given section point depends on the element type. *Figure 2* shows the integration points for a fully integrated element.

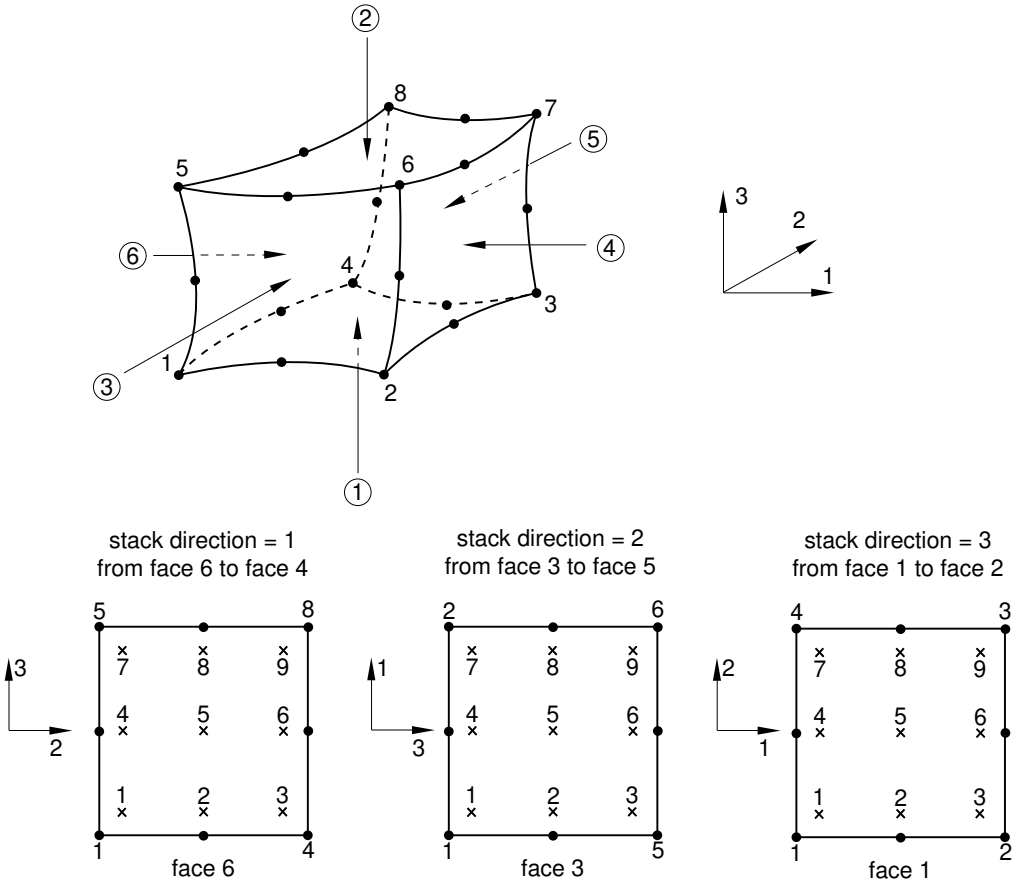


Figure 2: Stacking direction and associated element faces and positions of element integration point output variables in the layer plane.

The element faces are defined by the order in which the nodes are specified when the element is defined.

The element matrices are obtained by numerical integration. Gauss quadrature is used in the plane of the lamina, and Simpson's rule is used in the stacking direction. If one section point through the layer is used, it will be located in the middle of the layer thickness. The location of the section points in the plane of the lamina coincides with the location of the integration points. The number of section points required for the integration through the thickness of each layer is specified as part of the solid section definition; this number must be an odd number. The integration points for a fully integrated second-order composite element are shown in *Figure 2*, and the numbering of section points that are associated with an arbitrary integration point in a composite solid element is illustrated in *Figure 3*.

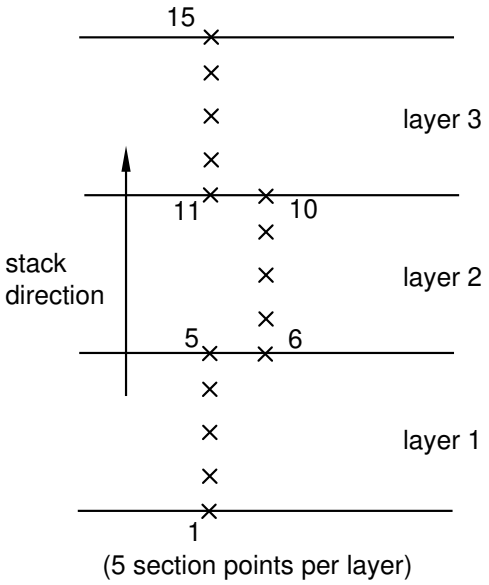


Figure 3: Numbering of section points in a three-layered composite element.

The thickness of each layer might not be constant from integration point to integration point within an element since the element dimensions in the stack direction might vary. Therefore, it is defined indirectly by specifying the ratio between the thickness and the element length along the stack direction in the solid section definition, as shown in *Figure 4*. Using the ratios that are defined for all layers, actual thicknesses will be determined at each integration point such that their sum equals the element length in the stack direction. The thickness ratios for the layers need not reflect actual element or model dimensions.

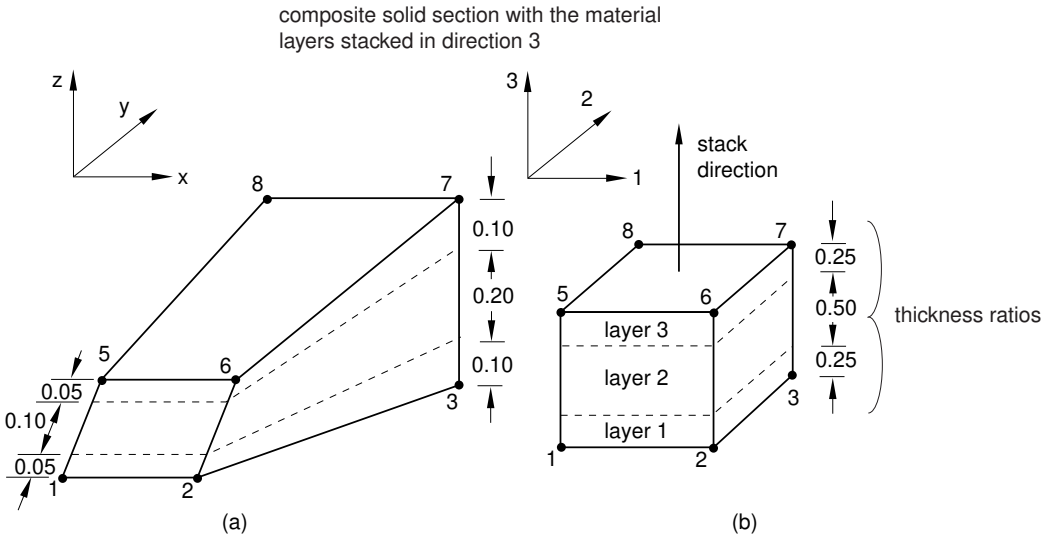


Figure 4: Lamina in (a) real space and (b) isoparametric space.

Unless your model is relatively simple, you will find it increasingly difficult to define your model using composite solid sections as you increase the number of layers and as you assign different sections to different regions. It can also be cumbersome to redefine the sections after you add new layers or remove or reposition existing layers. To manage a large number of layers in a typical composite model, you may want to use the composite layup functionality in Abaqus/CAE. For more information, see *Composite layups*.

For postprocessing composite solid elements appear in the output database (`.odb`) file with C1, C2, or C3 appended to the element type to represent the stacking direction of 1, 2, or 3, respectively.

Input File Usage: **SOLID SECTION*, COMPOSITE, STACK DIRECTION=1, 2, or 3, ELSET=*name*
thickness, number of integration points, material name, orientation name

Abaqus/CAE Usage: Abaqus/CAE uses a composite layup or a composite solid section to define the layers of a composite solid.

Use the following option for a composite layup:

Property module: **Create Composite Layup**: select **Solid** as the **Element Type**: specify stacking direction, regions, thicknesses, number of integration points, materials, and orientations

Use the following options for a composite solid section:

Property module:
Create Section: select **Solid** as the section **Category** and **Composite** as the section **Type**
Assign->Material Orientation: select regions: **Use Default Orientation or Other Method**: **Stacking Direction**: **Element direction 1**, **Element direction 2**, **Element direction 3**, or **From orientation**
Assign->Section: select regions

Output Locations for Composite Solid Elements

You specify the location of the output variables in the plane of the lamina (layers) when you request output of element variables. For example, you can request values at the centroid of each layer. In addition, you specify the number of output points through the thickness of the layers by providing a list of the "section points." The default section points for the output are the first and the last section point corresponding to the bottom and the top face, respectively (see *Figure 3*). See *Element Output* and *Writing Element Output to the Output Database* for more information.

Modeling Thick Composites with Solid Elements in Abaqus/Standard

While laminated composite solids are typically modeled using shell elements, the following cases require three-dimensional brick elements with one or multiple brick elements per layer: when transverse shear effects are predominant; when the normal stress cannot be ignored; and when accurate interlaminar stresses are required, such as near localized regions of complex loading or geometry.

One case in which shell elements perform somewhat better than solid elements is in modeling the transverse shear stress through the thickness. The transverse shear stresses in solid elements usually do not vanish at the free surfaces of the structure and are usually discontinuous at layer interfaces. This deficiency might be present even if several elements are used in the discretization through the section thickness. Since the transverse shear stresses in thick shell elements are calculated by Abaqus on the basis of linear elasticity theory, such stresses are often better estimated by thick shell elements than by solid elements (see *Composite shells in cylindrical bending*).

Defining Pressure Loads on Continuum Elements

The convention used for pressure loading on a continuum element is that positive pressure is directed into the element; that is, it pushes on the element. In large-strain analyses special consideration is necessary for plane stress elements that are pressure loaded on their edges; this issue is discussed in *Distributed Loads*.

Using Solid Elements in a Rigid Body

All solid elements can be included in a rigid body definition. When solid elements are assigned to a rigid body, they are no longer deformable and their motion is governed by the motion of the rigid body reference node (see *Rigid Body Definition*).

Section properties for solid elements that are part of a rigid body must be defined to properly account for rigid body mass and rotary inertia. All associated material properties will be ignored except for the density. Element output is not available for solid elements assigned to a rigid body.

Automatic Conversion of Certain Element Types in Abaqus/Standard

Element types C3D20 and C3D15 are converted automatically to the corresponding variable node element types C3D27 and C3D15V, respectively, if they have faces that are part of the secondary surface in a node-to-surface contact pair (see *Generally Applicable Contact Controls in Abaqus/Standard*).

Special Considerations for Various Element Types in Abaqus/Standard

The following considerations should be acknowledged in the context of the stress/displacement, coupled temperature-displacement, and heat transfer elements in Abaqus/Standard.

Interpolation of Temperature and Field Variables in Stress/Displacement Elements

The value of temperatures at the integration points used to compute the thermal stresses depends on whether first-order or second-order elements are used. An average temperature is used at the integration points in (compatible) linear elements so that the thermal strain is constant throughout the element; in the case of elements with incompatible modes the temperatures are interpolated linearly. An approximate linearly varying temperature distribution is used in higher-order elements with full integration. Higher-order reduced-integration elements pose no special problems since the temperatures are interpolated linearly. Field variables in a given stress/displacement element are interpolated using the same scheme used to interpolate temperatures.

Interpolation in Coupled Temperature-Displacement Elements

Coupled temperature-displacement elements use either linear or parabolic interpolation for the geometry and displacements. Temperature is interpolated linearly, but certain rules can apply to the temperature and field variable evaluation at the Gauss points, as discussed below.

Elements Using Linear Interpolation for Displacements and Temperatures

The elements that use linear interpolation for displacements and temperatures have temperatures at all nodes. The value of temperatures at the integration points used to compute the thermal stresses depends on whether incompatible mode elements are used. The thermal strain is taken as constant throughout the (compatible) element because it is desirable to have the same interpolation for thermal strains as for total strains so as to avoid spurious hydrostatic stresses. For elements with incompatible modes, the temperatures are interpolated linearly. Separate integration schemes are used for the internal energy storage, heat conduction, and plastic dissipation (coupling contribution) terms for the first-order elements. The internal energy storage term is integrated at the nodes, which yields a lumped internal energy matrix and, thereby, improves the accuracy for problems with latent heat effects. In fully integrated elements both the heat conduction and plastic dissipation terms are integrated at the Gauss points. While the plastic dissipation term is integrated at each Gauss point, the heat generated by the mechanical deformation at a Gauss point is applied at the nearest node. The temperature at a Gauss point is assumed to be the temperature of its nearest node to be consistent with the temperature treatment throughout the formulation. In reduced-integration elements the plastic dissipation term is obtained at the centroid and the heat generated by the mechanical deformation is applied as a weighted average at each node. The temperature at the centroid of reduced-integration elements is a weighted average of the nodal temperatures to be consistent with the temperature treatment throughout the formulation.

Elements Using Parabolic Interpolation for Displacements and Linear Interpolation for Temperatures

The elements that use parabolic interpolation for displacements and linear interpolation for temperatures have displacement degrees of freedom at all the nodes, but temperature degrees of freedom exist only at the corner nodes. The temperatures are interpolated linearly so that the thermal strains have the same interpolation as the total strains. Temperatures at the midside nodes are calculated by linear interpolation from the corner nodes for output purposes only. In contrast to the linear coupled elements, all terms in the governing equations are integrated using a conventional Gauss scheme. For these elements the stiffness matrix can be generated using either full integration (3 Gauss points in each parametric direction) or reduced integration (2 Gauss points in each parametric direction). The same integration scheme is always used for the specific heat and conductivity matrices as for the stiffness matrix; however, because of the lower-order interpolation for temperature, this implies that we always use a full integration scheme for the heat transfer matrices, even when the stiffness integration is reduced. Reduced integration uses a lower-order integration to form the element stiffness: the mass matrix and distributed loadings are still integrated exactly. Reduced integration usually provides more accurate results (providing that the elements are not distorted) and significantly reduces running time, especially in three dimensions. Reduced integration for the quadratic displacement elements is recommended in all cases except when very sharp strain gradients are expected (such as in finite-strain metal forming applications); these elements are considered to be the most cost-effective elements of this class.

Interpolation Scheme for Field Variables

The value of field variables at the integration points depends on whether first-order or second-order coupled temperature-displacement elements are used. An average field variable is used at the integration points in linear (compatible) elements. In the case of elements with incompatible modes the field variables are interpolated linearly. An approximate linearly varying field variable distribution is used in higher-order elements with full integration. Higher-order reduced-integration elements pose no special problems since the field variables are interpolated linearly.

Interpolation Scheme for Modified Elements

Modified triangle and tetrahedron elements use a special consistent interpolation scheme for displacement and temperature. Displacement and temperature degrees of freedom are active at all user-defined nodes.

Integration in Diffusive Heat Transfer Elements

In all the first-order elements (2-node links, 3-node triangles, 4-node quadrilaterals, 4-node tetrahedra, 6-node triangular prisms, and 8-node bricks) the internal energy storage term (associated with specific heat and latent heat storage) is integrated at the nodes. This integration scheme gives a diagonal internal energy matrix and improves the accuracy for problems with latent heat effects. Conduction contributions in these elements and all contributions in second-order elements use conventional Gauss schemes. Second-order elements are preferable for smooth problems without latent heat effects.

The one-dimensional element cannot be used in a mass diffusion analysis.

Forced Convection Heat Transfer Elements

These elements are available with linear interpolation only. They use an “upwinding” (Petrov-Galerkin) method to provide accurate solutions for convection-dominated problems (see *Convection/diffusion*). Consequently, the internal energy (associated with specific heat storage) is not integrated at the nodes, which yields a consistent internal energy matrix and might cause oscillatory temperatures if strong temperature gradients occur along boundaries that are parallel to the flow direction.

Electromagnetic Elements

These elements are available with linear edge-based interpolation only. The user-defined nodes define the geometry of the element but do not directly participate in the interpolation of the electromagnetic or, in the case of a magnetostatic analysis, the magnetic fields. However, temperature and predefined field variables are defined at the user-defined nodes and are interpolated to the integration points for evaluating material properties that are temperature and predefined field variable dependent.

Poroelastic Acoustic Elements

These elements are available with linear interpolation only for displacements and pore pressure. These elements are supported only in a direct steady-state dynamic procedure. The mass matrix of these elements is lumped only. There is no energy contribution from these elements.

Using Element Types C3D6 and C3D6T in Abaqus/Explicit Analyses

When element typesC3D6 and C3D6T are used in Abaqus/Explicit analyses, they appear in the output database (`.odb`) file as C3D6R and C3D6RT, respectively. In the data (`.dat`) file, C3D6 is referred to as C3D6R. You cannot specify C3D6R or C3D6RT as an element type for input.

References

- Vu-Quoc, L., and X. G. Tan, “Optimal Solid Shells for Non-Linear Analyses of Multilayer Composites. I. Statics,” *Computer Methods in AppliedMechanics and Engineering*, vol. 192, pp. 975–1016, 2003.

One-Dimensional Solid (Link) Element Library

Products: Abaqus/Standard Abaqus/CAE

Overview

This section provides a reference to the one-dimensional solid (link) elements available in Abaqus/Standard. For structural link (truss) elements, refer to *Truss Elements*.

References:

- *Solid (Continuum) Elements*
- **SOLID SECTION*

Element Types

Diffusive Heat Transfer Elements

DC1D2	2-node link
DC1D3	3-node link

Active Degrees of Freedom

11

Additional Solution Variables

None.

Forced Convection Heat Transfer Elements

DCC1D2	2-node link
DCC1D2D	2-node link with dispersion control

Active Degrees of Freedom

11

Additional Solution Variables

None.

Coupled Thermal-Electrical Elements

DC1D2E	2-node link
DC1D3E	3-node link

Active Degrees of Freedom

9, 11

Additional Solution Variables

None.

Acoustic Elements

AC1D2	2-node link
AC1D3	3-node link

Active Degrees of Freedom

8

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

You must provide the cross-sectional area of the element; by default, unit area is assumed.

Input File Usage: **SOLID SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Beam** as the section **Category** and **Truss** as the section **Type**

Element-Based Loading

Distributed Heat Fluxes

Distributed heat fluxes are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DFLUX): BF
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Heat body flux per unit volume.

Load ID (*DFLUX): BFNU
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Nonuniform heat body flux per unit volume with magnitude supplied via user subroutine *DFLUX*.

Load ID (*DFLUX): S1
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Heat surface flux per unit area into the first end of the link (node 1).

Load ID (*DFLUX): S2
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Heat surface flux per unit area into the second end of the link (node 2 or node 3).

Load ID (*DFLUX): S1NU
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}$
Description: Nonuniform heat surface flux per unit area into the first end of the link (node 1) with magnitude supplied via user subroutine *DFLUX*.

Load ID (*DFLUX): S2NU
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}$
Description: Nonuniform heat surface flux per unit area into the second end of the link (node 2 or node 3) with magnitude supplied via user subroutine *DFLUX*.

Film Conditions

Film conditions are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*FILM): F1
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\boldsymbol{\theta}^{-1}$
Description: Film coefficient and sink temperature (units of $\boldsymbol{\theta}$) at the first end of the link (node 1).

Load ID (*FILM): F2
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\boldsymbol{\theta}^{-1}$
Description: Film coefficient and sink temperature (units of $\boldsymbol{\theta}$) at the second end of the link (node 2 or node 3).

Load ID (*FILM): F1NU
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\boldsymbol{\theta}^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of θ) at the first end of the link (node 1) with magnitude supplied via user subroutine *FILM*.

Load ID (*FILM): F2NU
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of θ) at the second end of the link (node 2 or node 3) with magnitude supplied via user subroutine *FILM*.

Radiation Types

Radiation conditions are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*RADIATE): R1
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) at the first end of the link (node 1).

Load ID (*RADIATE): R2
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) at the second end of the link (node 2 or node 3).

Distributed Impedances

Distributed impedances are available for elements with acoustic pressure degrees of freedom. They are specified as described in *Acoustic and Shock Loads*.

Load ID (*IMPEDANCE): I1
Abaqus/CAE Load/Interaction: Not supported
Units: None
Description: Name of the impedance property that defines the impedance at the first end of the link (node 1).

Load ID (*IMPEDANCE): I2
Abaqus/CAE Load/Interaction: Not supported
Units: None
Description: Name of the impedance property that defines the impedance at the second end of the link (node 2 or node 3).

Distributed Electric Current Densities

Distributed electric current densities are available for coupled thermal-electrical elements. They are specified as described in *Coupled Thermal-Electrical Analysis*.

Load ID (*DECURRENT): CBF
Abaqus/CAE Load/Interaction: Body current
Units: $CL^{-3}T^{-1}$
Description: Volumetric current source density.

Load ID (*DECURRENT): CBFNU
Units: $CL^{-3}T^{-1}$

Description: Nonuniform volumetric current source density with magnitude supplied via user subroutine *DECURRENT*.

Load ID (*DECURRENT): CS1
Abaqus/CAE Load/Interaction: Surface current
Units: $CL^{-2}T^{-1}$
Description: Current density at the first end of the link (node 1).

Load ID (*DECURRENT): CS2
Abaqus/CAE Load/Interaction: Surface current
Units: $CL^{-2}T^{-1}$
Description: Current density at the second end of the link (node 2 or node 3).

Load ID (*DECURRENT): CS1NU
Abaqus/CAE Load/Interaction: Surface current
Units: $CL^{-2}T^{-1}$
Description: Nonuniform current density at the first end of the link (node 1) supplied via user subroutine *DECURRENT*.

Load ID (*DECURRENT): CS2NU
Abaqus/CAE Load/Interaction: Surface current
Units: $CL^{-2}T^{-1}$
Description: Nonuniform current density at the second end of the link (node 2 or node 3) supplied via user subroutine *DECURRENT*.

Element Output

Heat Flux Components

Available for elements with temperature degrees of freedom.

HFL1 Heat flux along the element axis.

Electrical Potential Gradient

Available for coupled thermal-electrical elements.

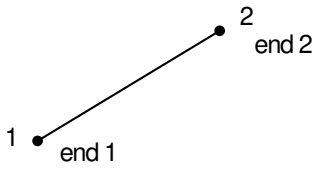
EPG1 Electrical potential gradient along the element axis.

Electrical Current Density Components

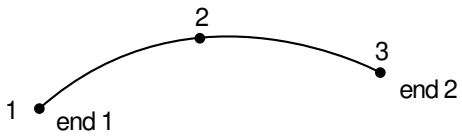
Available for coupled thermal-electrical elements.

ECD1 Electrical current density along the element axis.

Node Ordering and Face Numbering on Elements

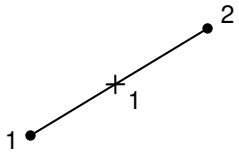


2 - node element

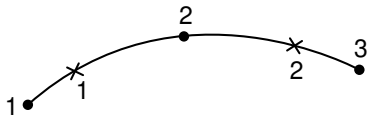


3 - node element

Numbering of Integration Points for Output



2 - node element



3 - node element

Two-Dimensional Solid Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the two-dimensional solid elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *Solid (Continuum) Elements*
- **SOLID SECTION*

Element Types

Plane Strain Elements

CPE3	3-node linear
CPE3H$^{(S)}$	3-node linear, hybrid with constant pressure
CPE4$^{(S)}$	4-node bilinear
CPE4H$^{(S)}$	4-node bilinear, hybrid with constant pressure
CPE4I$^{(S)}$	4-node bilinear, incompatible modes
CPE4IH$^{(S)}$	4-node bilinear, incompatible modes, hybrid with linear pressure
CPE4R	4-node bilinear, reduced integration with hourglass control
CPE4RH$^{(S)}$	4-node bilinear, reduced integration with hourglass control, hybrid with constant pressure
CPE6$^{(S)}$	6-node quadratic
CPE6H$^{(S)}$	6-node quadratic, hybrid with linear pressure
CPE6M	6-node modified plane strain triangle, hourglass control
CPE6MH$^{(S)}$	6-node modified, with hourglass control, hybrid with linear pressure
CPE8$^{(S)}$	8-node biquadratic
CPE8H$^{(S)}$	8-node biquadratic, hybrid with linear pressure
CPE8R$^{(S)}$	8-node biquadratic, reduced integration
CPE8RH$^{(S)}$	8-node biquadratic, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2

Additional Solution Variables

The constant pressure hybrid elements have one additional variable relating to pressure, and the linear pressure hybrid elements have three additional variables relating to pressure.

Element types CPE4I and CPE4IH have five additional variables relating to the incompatible modes.

Element types CPE6M and CPE6MH have two additional displacement variables.

Plane Stress Elements

CPS3	3-node linear
CPS4$^{(S)}$	4-node bilinear
CPS4I$^{(S)}$	4-node bilinear, incompatible modes
CPS4R	4-node bilinear, reduced integration with hourglass control
CPS6$^{(S)}$	6-node quadratic
CPS6M	6-node modified plane stress triangle, hourglass control
CPS8$^{(S)}$	8-node biquadratic
CPS8R$^{(S)}$	8-node biquadratic, reduced integration

Active Degrees of Freedom

1, 2

Additional Solution Variables

Element type CPS4I has four additional variables relating to the incompatible modes.

Element type CPS6M has two additional displacement variables.

Generalized Plane Strain Elements

CPEG3$^{(S)}$	3-node linear triangle
CPEG3H$^{(S)}$	3-node linear triangle, hybrid with constant pressure
CPEG4$^{(S)}$	4-node bilinear quadrilateral
CPEG4H$^{(S)}$	4-node bilinear quadrilateral, hybrid with constant pressure
CPEG4I$^{(S)}$	4-node bilinear quadrilateral, incompatible modes
CPEG4IH$^{(S)}$	4-node bilinear quadrilateral, incompatible modes, hybrid with linear pressure
CPEG4R$^{(S)}$	4-node bilinear quadrilateral, reduced integration with hourglass control
CPEG4RH$^{(S)}$	4-node bilinear quadrilateral, reduced integration with hourglass control, hybrid with constant pressure

CPEG6[(S)]	6-node quadratic triangle
CPEG6H[(S)]	6-node quadratic triangle, hybrid with linear pressure
CPEG6M[(S)]	6-node modified, with hourglass control
CPEG6MH[(S)]	6-node modified, with hourglass control, hybrid with linear pressure
CPEG8[(S)]	8-node biquadratic quadrilateral
CPEG8H[(S)]	8-node biquadratic quadrilateral, hybrid with linear pressure
CPEG8R[(S)]	8-node biquadratic quadrilateral, reduced integration
CPEG8RH[(S)]	8-node biquadratic quadrilateral, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2 at all but the reference node

3, 4, 5 at the reference node

Additional Solution Variables

The constant pressure hybrid elements have one additional variable relating to pressure, and the linear pressure hybrid elements have three additional variables relating to pressure.

Element types CPEG4I and CPEG4IH have five additional variables relating to the incompatible modes.

Element types CPEG6M and CPEG6MH have two additional displacement variables.

Coupled Temperature-Displacement Plane Strain Elements

CPE3T	3-node linear displacement and temperature
CPE4T[(S)]	4-node bilinear displacement and temperature
CPE4HT[(S)]	4-node bilinear displacement and temperature, hybrid with constant pressure
CPE4RT	4-node bilinear displacement and temperature, reduced integration with hourglass control
CPE4RHT[(S)]	4-node bilinear displacement and temperature, reduced integration with hourglass control, hybrid with constant pressure
CPE6MT	6-node modified plane strain thermally coupled triangle, hourglass control
CPE6MHT[(S)]	6-node modified displacement and temperature, with hourglass control, hybrid with constant pressure
CPE8T[(S)]	8-node biquadratic displacement, bilinear temperature
CPE8HT[(S)]	8-node biquadratic displacement, bilinear temperature, hybrid with linear pressure
CPE8RT[(S)]	8-node biquadratic displacement, bilinear temperature, reduced integration
CPE8RHT[(S)]	8-node biquadratic displacement, bilinear temperature, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2, 11 at corner nodes

1, 2 at midside nodes of second-order elements in Abaqus/Standard

1, 2, 11 at midside nodes of modified displacement and temperature elements in Abaqus/Standard

Additional Solution Variables

The constant pressure hybrid elements have one additional variable relating to pressure, and the linear pressure hybrid elements have three additional variables relating to pressure.

Element types CPE6MT and CPE6MHT have two additional displacement variables and one additional temperature variable.

Coupled Temperature-Displacement Plane Stress Elements

CPS3T	3-node linear displacement and temperature
CPS4T$^{(S)}$	4-node bilinear displacement and temperature
CPS4RT	4-node bilinear displacement and temperature, reduced integration with hourglass control
CPS6MT	6-node modified plane stress thermally coupled triangle, hourglass control
CPS8T$^{(S)}$	8-node biquadratic displacement, bilinear temperature
CPS8RT$^{(S)}$	8-node biquadratic displacement, bilinear temperature, reduced integration

Active Degrees of Freedom

1, 2, 11 at corner nodes

1, 2 at midside nodes of second-order elements in Abaqus/Standard

1, 2, 11 at midside nodes of modified displacement and temperature elements in Abaqus/Standard

Additional Solution Variables

Element type CPS6MT has two additional displacement variables and one additional temperature variable.

Coupled Temperature-Displacement Generalized Plane Strain Elements

CPEG3T$^{(S)}$	3-node linear displacement and temperature
CPEG3HT$^{(S)}$	3-node linear displacement and temperature, hybrid with constant pressure
CPEG4T$^{(S)}$	4-node bilinear displacement and temperature
CPEG4HT$^{(S)}$	4-node bilinear displacement and temperature, hybrid with constant pressure

CPEG4RT$^{(S)}$	4-node bilinear displacement and temperature, reduced integration with hourglass control
CPEG4RHT$^{(S)}$	4-node bilinear displacement and temperature, reduced integration with hourglass control, hybrid with constant pressure
CPEG6MT$^{(S)}$	6-node modified displacement and temperature, with hourglass control
CPEG6MHT$^{(S)}$	6-node modified displacement and temperature, with hourglass control, hybrid with constant pressure
CPEG8T$^{(S)}$	8-node biquadratic displacement, bilinear temperature
CPEG8HT$^{(S)}$	8-node biquadratic displacement, bilinear temperature, hybrid with linear pressure
CPEG8RHT$^{(S)}$	8-node biquadratic displacement, bilinear temperature, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2, 11 at corner nodes

1, 2 at midside nodes of second-order elements

1, 2, 11 at midside nodes of modified displacement and temperature elements

3, 4, 5 at the reference node

Additional Solution Variables

The constant pressure hybrid elements have one additional variable relating to pressure, and the linear pressure hybrid elements have three additional variables relating to pressure.

Element types CPEG6MT and CPEG6MHT have two additional displacement variables and one additional temperature variable.

Diffusive Heat Transfer or Mass Diffusion Elements

DC2D3$^{(S)}$	3-node linear
DC2D4$^{(S)}$	4-node linear
DC2D6$^{(S)}$	6-node quadratic
DC2D8$^{(S)}$	8-node biquadratic

Active Degrees of Freedom

11

Additional Solution Variables

None.

Forced Convection/Diffusion Elements

DCC2D4[(S)]	4-node
DCC2D4D[(S)]	4-node with dispersion control

Active Degrees of Freedom

11

Additional Solution Variables

None.

Coupled Thermal-Electrical Elements

DC2D3E[(S)]	3-node linear
DC2D4E[(S)]	4-node linear
DC2D6E[(S)]	6-node quadratic
DC2D8E[(S)]	8-node biquadratic

Active Degrees of Freedom

9, 11

Additional Solution Variables

None.

Pore Pressure Plane Strain Elements

CPE4P[(S)]	4-node bilinear displacement and pore pressure
CPE4PH[(S)]	4-node bilinear displacement and pore pressure, hybrid with constant pressure stress
CPE4RP[(S)]	4-node bilinear displacement and pore pressure, reduced integration with hourglass control
CPE4RPH[(S)]	4-node bilinear displacement and pore pressure, reduced integration with hourglass control, hybrid with constant pressure
CPE6MP[(S)]	6-node modified displacement and pore pressure, with hourglass control
CPE6MPH[(S)]	6-node modified displacement and pore pressure, with hourglass control, hybrid with linear pressure
CPE8P[(S)]	8-node biquadratic displacement, bilinear pore pressure

CPE8PH[(S)]	8-node biquadratic displacement, bilinear pore pressure, hybrid with linear pressure stress
CPE8RP[(S)]	8-node biquadratic displacement, bilinear pore pressure, reduced integration
CPE8RPH[(S)]	8-node biquadratic displacement, bilinear pore pressure, reduced integration, hybrid with linear pressure stress

Active Degrees of Freedom

1, 2, 8 at corner nodes

1, 2 at midside nodes for all elements except CPE6MP and CPE6MPH, which also have degree of freedom 8 active at midside nodes

Additional Solution Variables

The constant pressure hybrid elements have one additional variable relating to the effective pressure stress, and the linear pressure hybrid elements have three additional variables relating to the effective pressure stress to permit fully incompressible material modeling.

Element types CPE6MP and CPE6MPH have two additional displacement variables and one additional pore pressure variable.

Coupled Temperature–Pore Pressure Plane Strain Elements

CPE4PT[(S)]	4-node bilinear displacement, pore pressure, and temperature
CPE4PHT[(S)]	4-node bilinear displacement, pore pressure, and temperature; hybrid with constant pressure stress
CPE4RPT[(S)]	4-node bilinear displacement, pore pressure, and temperature; reduced integration
CPE4RPHT[(S)]	4-node bilinear displacement, pore pressure, and temperature; reduced integration, hybrid with constant pressure stress

Active Degrees of Freedom

1, 2, 8, 11 at corner nodes

Additional Solution Variables

The constant pressure stress hybrid elements have one additional variable relating to the effective pressure stress to permit fully incompressible material modeling.

Acoustic Elements

AC2D3	3-node linear
AC2D4[(S)]	4-node bilinear

AC2D4R$^{(E)}$	4-node bilinear, reduced integration with hourglass control
AC2D6$^{(S)}$	6-node quadratic
AC2D8$^{(S)}$	8-node biquadratic

Active Degrees of Freedom

8

Additional Solution Variables

None.

Piezoelectric Plane Strain Elements

CPE3E$^{(S)}$	3-node linear
CPE4E$^{(S)}$	4-node bilinear
CPE6E$^{(S)}$	6-node quadratic
CPE8E$^{(S)}$	8-node biquadratic
CPE8RE$^{(S)}$	8-node biquadratic, reduced integration

Active Degrees of Freedom

1, 2, 9

Additional Solution Variables

None.

Piezoelectric Plane Stress Elements

CPS3E$^{(S)}$	3-node linear
CPS4E$^{(S)}$	4-node bilinear
CPS6E$^{(S)}$	6-node quadratic
CPS8E$^{(S)}$	8-node biquadratic
CPS8RE$^{(S)}$	8-node biquadratic, reduced integration

Active Degrees of Freedom

1, 2, 9

Additional Solution Variables

None.

Electromagnetic Elements

EMC2D3$^{(S)}$	3-node zero-order
EMC2D4$^{(S)}$	4-node zero-order

Active Degrees of Freedom

Magnetic vector potential (for more information, see *Boundary Conditions* and *Boundary Conditions*).

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*

Element Property Definition

For all elements except generalized plane strain elements, you must provide the element thickness; by default, unit thickness is assumed.

For generalized plane strain elements, you must provide three values: the initial length of the axial material fiber through the reference node, the initial value of $\Delta\phi_x$ (in radians), and the initial value of $\Delta\phi_y$ (in radians). If you do not provide these values, Abaqus assumes the default values of one unit as the initial length and zero for $\Delta\phi_x$ and $\Delta\phi_y$. In addition, you must define the reference point for generalized plane strain elements.

Input File Usage: Use the following option to define the element properties for all elements except generalized plane strain elements:

**SOLID SECTION*

Use the following option to define the element properties for generalized plane strain elements:

**SOLID SECTION*, REF NODE=*node number or node set name*

Abaqus/CAE Usage: Property module: **Create Section**: select **Solid** as the section **Category** and **Homogeneous**, **Generalized plane strain**, or **Electromagnetic, Solid** as the section **Type**

Generalized plane strain sections must be assigned to regions of parts that have a reference point associated with them. To define the reference point:

Part module: **Tools->Reference Point**: select reference point

Element-Based Loading

Distributed Loads

Distributed loads are available for all elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *X*-direction.

Load ID (*DLOAD): BY
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *Y*-direction.

Load ID (*DLOAD): BXNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *X*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BYNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *Y*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): CENT (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}(ML^{-3}T^{-2})$

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume, ω is the angular velocity). Not available for pore pressure elements.

Load ID (*DLOAD): CENTRIF (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): CORIO (S)
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-4}T$ $(ML^{-3}T^{-1})$

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit volume, ω is the angular velocity). Not available for pore pressure elements.

Load ID (*DLOAD): GRAV

Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): HP*n* (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Hydrostatic pressure on face *n*, linear in global *Y*.

Load ID (*DLOAD): P*n*
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on face *n*.

Load ID (*DLOAD): P*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure on face *n* with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): ROTA (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}
Description: Rotary acceleration load (magnitude is input as $\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is the rotary acceleration).

Load ID (*DLOAD): SBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-5}T^2$
Description: Stagnation body force in global *X*- and *Y*-directions.

Load ID (*DLOAD): SP*n* (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T^2$
Description: Stagnation pressure on face *n*.

Load ID (*DLOAD): TRSHR*n*
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on face *n*.

Load ID (*DLOAD): TRSHR*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform shear traction on face *n* with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC*n*
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on face *n*.

Load ID (*DLOAD): TRVEC*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported

Units: FL^{-2}

Description: Nonuniform general traction on face *n* with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): VBF (E)

Abaqus/CAE Load/Interaction: Not supported

Units: $FL^{-4}T$

Description: Viscous body force in global *X*- and *Y*-directions.

Load ID (*DLOAD): VP*n* (E)

Abaqus/CAE Load/Interaction: Not supported

Units: $FL^{-3}T$

Description: Viscous pressure on face *n*, applying a pressure proportional to the velocity normal to the face and opposing the motion.

Foundations

Foundations are available for Abaqus/Standard elements with displacement degrees of freedom. They are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): F*n* (S)

Abaqus/CAE Load/Interaction: Elastic foundation

Units: FL^{-3}

Description: Elastic foundation on face *n*.

Distributed Heat Fluxes

Distributed heat fluxes are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DFLUX): BF

Abaqus/CAE Load/Interaction: Body heat flux

Units: $JL^{-3}T^{-1}$

Description: Heat body flux per unit volume.

Load ID (*DFLUX): BFNU

Abaqus/CAE Load/Interaction: Body heat flux

Units: $JL^{-3}T^{-1}$

Description: Nonuniform heat body flux per unit volume with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard and *VDFLUX* in Abaqus/Explicit.

Load ID (*DFLUX): S*n*

Abaqus/CAE Load/Interaction: Surface heat flux

Units: $JL^{-2}T^{-1}$

Description: Heat surface flux per unit area into face *n*.

Load ID (*DFLUX): S*n*NU

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}T^{-1}$

Description: Nonuniform heat surface flux per unit area into face *n* with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard and *VDFLUX* in Abaqus/Explicit.

Film Conditions

Film conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*FILM): F*n*
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Film coefficient and sink temperature (units of θ) provided on face *n*.

Load ID (*FILM): F*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of θ) provided on face *n* with magnitude supplied via user subroutine *FILM*.

Radiation Types

Radiation conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*RADIATE): R*n*
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) provided on face *n*.

Distributed Flows

Distributed flows are available for all elements with pore pressure degrees of freedom. They are specified as described in *Pore Fluid Flow*.

Load ID (*FLOW): Q*n* (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^{3}T^{-1}$

Description: Seepage coefficient and reference sink pore pressure (units of FL^{-2}) provided on face *n*.

Load ID (*FLOW): Q*n*D (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^{3}T^{-1}$

Description: Drainage-only seepage coefficient provided on face *n*.

Load ID (*FLOW): Q*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^{3}T^{-1}$

Description: Nonuniform seepage coefficient and reference sink pore pressure (units of FL^{-2}) provided on face *n* with magnitude supplied via user subroutine *FLOW*.

Load ID (*DFLOW): S*n* (S)
Abaqus/CAE Load/Interaction: Surface pore fluid
Units: LT^{-1}

Description: Prescribed pore fluid effective velocity (outward from the face) on face *n*.

Load ID (*DFLOW): S*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: LT^{-1}
Description: Nonuniform prescribed pore fluid effective velocity (outward from the face) on face *n* with magnitude supplied via user subroutine *DFLOW*.

Distributed Impedances

Distributed impedances are available for all elements with acoustic pressure degrees of freedom. They are specified as described in *Acoustic and Shock Loads*.

Load ID (*IMPEDANCE): I*n*
Abaqus/CAE Load/Interaction: Not supported
Units: None
Description: Name of the impedance property that defines the impedance on face *n*.

Electric Fluxes

Electric fluxes are available for piezoelectric elements. They are specified as described in *Piezoelectric Analysis*.

Load ID (*DECHARGE): EBF (S)
Abaqus/CAE Load/Interaction: Body charge
Units: CL^{-3}
Description: Body flux per unit volume.

Load ID (*DECHARGE): ES*n* (S)
Abaqus/CAE Load/Interaction: Surface charge
Units: CL^{-2}
Description: Prescribed surface charge on face *n*.

Distributed Electric Current Densities

Distributed electric current densities are available for coupled thermal-electrical elements, coupled thermal-electrical-structural elements, and electromagnetic elements. They are specified as described in *Coupled Thermal-Electrical Analysis*, *Fully Coupled Thermal-Electrical-Structural Analysis*, and *Eddy Current Analysis*.

Load ID (*DECURRENT): CBF (S)
Abaqus/CAE Load/Interaction: Body current
Units: $CL^{-3}T^{-1}$
Description: Volumetric current source density.

Load ID (*DECURRENT): CBFNU
Units: $CL^{-3}T^{-1}$
Description: Nonuniform volumetric current source density with magnitude supplied via user subroutine *DECURRENT*.

Load ID (*DECURRENT): CS*n* (S)
Abaqus/CAE Load/Interaction: Surface current
Units: $CL^{-2}T^{-1}$
Description: Current density on face *n*.

Load ID (*DECURRENT): CS*n*NU (S)
Units: $CL^{-2}T^{-1}$

Description: Nonuniform current density on face *n* with magnitude supplied via user subroutine *DECURRENT*.

Load ID (*DECURRENT): CJ (S)
Abaqus/CAE Load/Interaction: Body current density
Units: $CL^{-2}T^{-1}$
Description: Volume current density vector in an eddy current analysis.

Distributed Concentration Fluxes

Distributed concentration fluxes are available for mass diffusion elements. They are specified as described in *Mass Diffusion Analysis*.

Load ID (*DFLUX): BF (S)
Abaqus/CAE Load/Interaction: Body concentration flux
Units: PT^{-1}
Description: Concentration body flux per unit volume.

Load ID (*DFLUX): BFNU (S)
Abaqus/CAE Load/Interaction: Body concentration flux
Units: PT^{-1}
Description: Nonuniform concentration body flux per unit volume with magnitude supplied via user subroutine *DFLUX*.

Load ID (*DFLUX): S*n* (S)
Abaqus/CAE Load/Interaction: Surface concentration flux
Units: PLT^{-1}
Description: Concentration surface flux per unit area into face *n*.

Load ID (*DFLUX): S*n*NU (S)
Abaqus/CAE Load/Interaction: Surface concentration flux
Units: PLT^{-1}
Description: Nonuniform concentration surface flux per unit area into face *n* with magnitude supplied via user subroutine *DFLUX*.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are available for all elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): HP (S)
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Hydrostatic pressure on the element surface, linear in global *Y*.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element surface.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure

Units: FL^{-2}

Description: Nonuniform pressure on the element surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DSLOAD): SP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-4}T^2$

Description: Stagnation pressure on the element surface.

Load ID (*DSLOAD): ST
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}

Description: Surface tension on the element surface for plane strain elements.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Shear traction on the element surface.

Load ID (*DSLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Nonuniform shear traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: General traction on the element surface.

Load ID (*DSLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Nonuniform general traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): VP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-3}T$

Description: Viscous pressure on the element surface. The viscous pressure is proportional to the velocity normal to the element surface and opposing the motion.

Distributed Heat Fluxes

Surface-based heat fluxes are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DSFLUX): S
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$

Description: Heat surface flux per unit area into the element surface.

Load ID (*DSFLUX): SNU

Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$

Description: Nonuniform heat surface flux per unit area applied on the element surface with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard and *VDFLUX* in Abaqus/Explicit.

Film Conditions

Surface-based film conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*SFILM): F
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\boldsymbol{\theta}^{-1}$

Description: Film coefficient and sink temperature (units of $\boldsymbol{\theta}$) provided on the element surface.

Load ID (*SFILM): FNU (S)
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\boldsymbol{\theta}^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of $\boldsymbol{\theta}$) provided on the element surface with magnitude supplied via user subroutine *FILM*.

Radiation Types

Surface-based radiation conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*SRADIATE): R
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature (units of $\boldsymbol{\theta}$) provided on the element surface.

Distributed Flows

Surface-based flows are available for all elements with pore pressure degrees of freedom. They are specified as described in *Pore Fluid Flow*.

Load ID (*SFLOW): Q (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^{3}T^{-1}$

Description: Seepage coefficient and reference sink pore pressure (units of FL^{-2}) provided on the element surface.

Load ID (*SFLOW): QD (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^{3}T^{-1}$

Description: Drainage-only seepage coefficient provided on the element surface.

Load ID (*SFLOW): QNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^{3}T^{-1}$

Description: Nonuniform seepage coefficient and reference sink pore pressure (units of FL^{-2}) provided on the element surface with magnitude supplied via user subroutine *FLOW*.

Load ID (*DSFLOW): S (S)
Abaqus/CAE Load/Interaction: Surface pore fluid
Units: LT^{-1}
Description: Prescribed pore fluid effective velocity outward from the element surface.

Load ID (*DSFLOW): SNU (S)
Abaqus/CAE Load/Interaction: Surface pore fluid
Units: LT^{-1}
Description: Nonuniform prescribed pore fluid effective velocity outward from the element surface with magnitude supplied via user subroutine *DFLOW*.

Distributed Impedances

Surface-based impedances are available for all elements with acoustic pressure degrees of freedom. They are specified as described in *Acoustic and Shock Loads*.

Incident Wave Loading

Surface-based incident wave loads are available for all elements with displacement degrees of freedom or acoustic pressure degrees of freedom. They are specified as described in *Acoustic and Shock Loads*. If the incident wave field includes a reflection off a plane outside the boundaries of the mesh, this effect can be included.

Electric Fluxes

Surface-based electric fluxes are available for piezoelectric elements. They are specified as described in *Piezoelectric Analysis*.

Load ID (*DSECHARGE): ES (S)
Abaqus/CAE Load/Interaction: Surface charge
Units: CL^{-2}
Description: Prescribed surface charge on the element surface.

Distributed Electric Current Densities

Surface-based electric current densities are available for coupled thermal-electrical elements, coupled thermal-electrical-structural elements, and electromagnetic elements. They are specified as described in *Coupled Thermal-Electrical Analysis*, *Fully Coupled Thermal-Electrical-Structural Analysis*, and *Eddy Current Analysis*.

Load ID (*DSECURRENT): CS (S)
Abaqus/CAE Load/Interaction: Surface current
Units: $CL^{-2}T^{-1}$
Description: Current density applied on the element surface.

Load ID (*DSECURRENT): CSNU (S)
Units: $CL^{-2}T^{-1}$
Description: Nonuniform current density on the element surface with magnitude supplied via user subroutine *DECURRENT*.

Load ID (*DSECURRENT): CK (S)
Abaqus/CAE Load/Interaction: Surface current density
Units: $CL^{-1}T^{-1}$
Description: Surface current density vector in an eddy current analysis.

Element Output

For most elements output is in global directions unless a local coordinate system is assigned to the element through the section definition (*Orientations*) in which case output is in the local coordinate system (which rotates with the motion in large-displacement analysis). See *State storage* for details.

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11 ***XX***, direct stress.

S22 ***YY***, direct stress.

S33 ***ZZ***, direct stress (not available for plane stress elements).

S12 ***XY***, shear stress.

Heat Flux Components

Available for elements with temperature degrees of freedom.

HFL1 Heat flux in the *X*-direction.

HFL2 Heat flux in the *Y*-direction.

Pore Fluid Velocity Components

Available for elements with pore pressure degrees of freedom.

FLVEL1 Pore fluid effective velocity in the *X*-direction.

FLVEL2 Pore fluid effective velocity in the *Y*-direction.

Mass Concentration Flux Components

Available for elements with normalized concentration degrees of freedom.

MFL1 Concentration flux in the *X*-direction.

MFL2 Concentration flux in the *Y*-direction.

Electrical Potential Gradient

Available for elements with electrical potential degrees of freedom.

EPG1 Electrical potential gradient in the *X*-direction.

EPG2 Electrical potential gradient in the *Y*-direction.

Electrical Flux Components

Available for piezoelectric elements.

EFLX1 Electrical flux in the *X*-direction.

EFLX2 Electrical flux in the *Y*-direction.

Electrical Current Density Components

Available for coupled thermal-electrical elements.

ECD1 Electrical current density in the *X*-direction.

ECD2 Electrical current density in the *Y*-direction.

Electrical Field Components

Available for electromagnetic elements in an eddy current analysis.

EME1 Electric field in the *X*-direction.

EME2 Electric field in the *Y*-direction.

Magnetic Flux Density Components

Available for electromagnetic elements.

EMB3 Magnetic flux density in the *Z*-direction.

Magnetic Field Components

Available for electromagnetic elements.

EMH3 Magnetic field in the *Z*-direction.

Eddy Current Density Components in an Eddy Current Analysis

Available for electromagnetic elements in an eddy current analysis.

EMCD1 Eddy current density in the *X*-direction.

EMCD2 Eddy current density in the *Y*-direction.

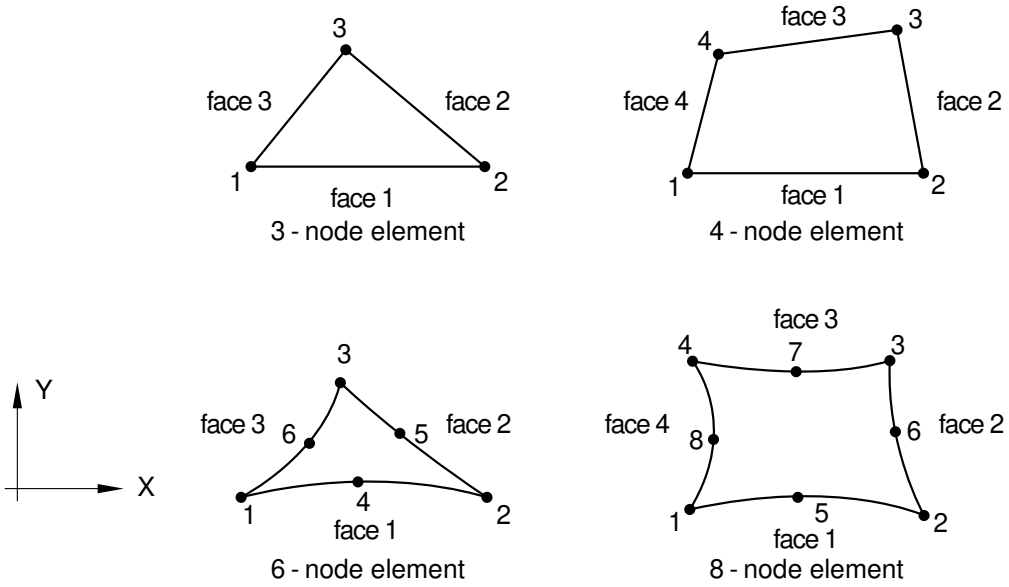
Applied Volume Current Density Components in an Eddy Current or Magnetostatic Analysis

Available for electromagnetic elements in an eddy current or magnetostatic analysis.

EMCDA1 Applied volume current density in the *X*-direction.

EMCDA2 Applied volume current density in the *Y*-direction.

Node Ordering and Face Numbering on Elements



For generalized plane strain elements, the reference node associated with each element (where the generalized plane strain degrees of freedom are stored) is not shown. The reference node should be the same for all elements in any given connected region so that the bounding planes are the same for that region. Different regions might have different reference nodes. The number of the reference node is not incremented when the elements are generated incrementally (see *Creating Elements from Existing Elements by Generating Them Incrementally*).

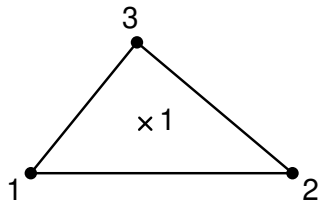
Table 1: Triangular element faces

Face 1	1 – 2 face
Face 2	2 – 3 face
Face 3	3 – 1 face

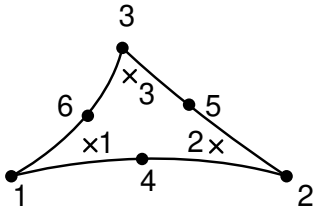
Table 2: Quadrilateral element faces

Face 1	1 – 2 face
Face 2	2 – 3 face
Face 3	3 – 4 face
Face 4	4 – 1 face

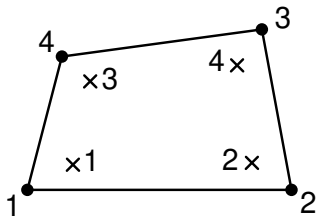
Numbering of Integration Points for Output



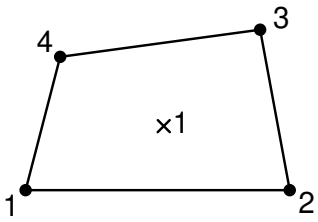
3 - node element



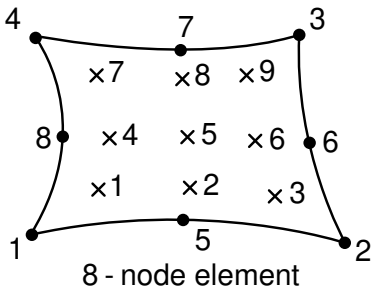
6 - node element



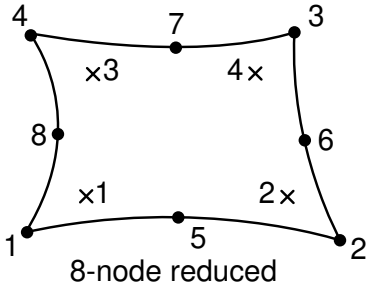
4 - node element



4-node reduced integration element



8 - node element



8-node reduced integration element

For heat transfer applications a different integration scheme is used for triangular elements, as described in *Triangular, tetrahedral, and wedge elements*.

Three-Dimensional Solid Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the three-dimensional solid elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *Solid (Continuum) Elements*
- **SOLID SECTION*

Element Types

Stress/Displacement Elements

C3D4	4-node linear tetrahedron
C3D4H	4-node linear tetrahedron, hybrid, linear pressure
C3D5	5-node linear pyramid
C3D5H$^{(S)}$	5-node linear pyramid, hybrid with constant pressure
C3D6$^{(S)}$	6-node linear wedge
C3D6$^{(E)}$	6-node linear wedge, reduced integration, hourglass control
C3D6H$^{(S)}$	6-node linear wedge, hybrid with constant pressure
C3D8	8-node linear brick
C3D8H$^{(S)}$	8-node linear brick, hybrid with constant pressure
C3D8I	8-node linear brick, incompatible modes
C3D8IH$^{(S)}$	8-node linear brick, incompatible modes, hybrid with linear pressure
C3D8R	8-node linear brick, reduced integration with hourglass control
C3D8RH$^{(S)}$	8-node linear brick, reduced integration with hourglass control, hybrid with constant pressure
C3D8S$^{(S)}$	8-node linear brick, improved surface stress visualization
C3D8HS$^{(S)}$	8-node linear brick, hybrid with constant pressure, improved surface stress visualization
C3D10	10-node quadratic tetrahedron
C3D10H$^{(S)}$	10-node quadratic tetrahedron, hybrid with constant pressure
C3D10HS$^{(S)}$	10-node general-purpose quadratic tetrahedron, hybrid with linear pressure, improved surface stress visualization
C3D10M	10-node modified tetrahedron, hourglass control

C3D10MH[(S)]	10-node modified tetrahedron, with hourglass control, hybrid with linear pressure
C3D15[(S)]	15-node quadratic wedge
C3D15H[(S)]	15-node quadratic wedge, hybrid with linear pressure
C3D20[(S)]	20-node quadratic brick
C3D20H[(S)]	20-node quadratic brick, hybrid with linear pressure
C3D20R[(S)]	20-node quadratic brick, reduced integration
C3D20RH[(S)]	20-node quadratic brick, reduced integration, hybrid with linear pressure
CSS8[(S)]	8-node linear solid shell brick, incompatible modes, with assumed strain

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

The constant pressure hybrid elements have one additional variable relating to pressure, and the linear pressure hybrid elements have four additional variables relating to pressure.

Element type C3D8I has thirteen additional variables relating to the incompatible modes, and element type C3D8IH has nine additional variables relating to the incompatible modes.

Element type C3D10HS has four additional variables relating to pressure that are activated automatically for a given element once the material exhibits behavior approaching the incompressible limit.

Element types C3D10M and C3D10MH have three additional displacement variables.

Element type CSS8 has seven additional variables relating to the incompatible modes.

Stress/Displacement Variable Node Elements

C3D15V[(S)]	15 to 18-node wedge
C3D15VH[(S)]	15 to 18-node wedge, hybrid with linear pressure
C3D27[(S)]	21 to 27-node brick
C3D27H[(S)]	21 to 27-node brick, hybrid with linear pressure
C3D27R[(S)]	21 to 27-node brick, reduced integration
C3D27RH[(S)]	21 to 27-node brick, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

The hybrid elements have four additional variables relating to pressure.

Coupled Temperature-Displacement Elements

C3D4T	4-node linear displacement and temperature
C3D6T$^{(S)}$	6-node thermally coupled wedge, linear displacement and temperature
C3D6T$^{(E)}$	6-node thermally coupled wedge, linear displacement and temperature, reduced integration, hourglass control
C3D6HT$^{(S)}$	6-node linear displacement and temperature, hybrid with constant pressure
C3D8T	8-node trilinear displacement and temperature
C3D8HT$^{(S)}$	8-node trilinear displacement and temperature, hybrid with constant pressure
C3D8RT	8-node trilinear displacement and temperature, reduced integration with hourglass control
C3D8RHT$^{(S)}$	8-node trilinear displacement and temperature, reduced integration with hourglass control, hybrid with constant pressure
C3D8IT	8-node linear brick, displacement and temperature, incompatible modes
C3D8IHT$^{(S)}$	8-node linear brick, displacement and temperature, hybrid with linear pressure, incompatible modes
C3D10T$^{(S)}$	10-node triquadratic displacement, trilinear temperature
C3D10T$^{(E)}$	10-node triquadratic displacement, triquadratic temperature
C3D10HT$^{(S)}$	10-node triquadratic displacement, trilinear temperature, hybrid with constant pressure
C3D10MT	10-node modified thermally coupled tetrahedron, hourglass control l
C3D10MHT$^{(S)}$	10-node modified displacement and temperature tetrahedron, with hourglass control, hybrid with linear pressure
C3D20T$^{(S)}$	20-node triquadratic displacement, trilinear temperature
C3D20HT$^{(S)}$	20-node triquadratic displacement, trilinear temperature, hybrid with linear pressure
C3D20RT$^{(S)}$	20-node triquadratic displacement, trilinear temperature, reduced integration
C3D20RHT$^{(S)}$	20-node triquadratic displacement, trilinear temperature, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2, 3, 11 at corner nodes

1, 2, 3 at midside nodes of second-order elements in Abaqus/Standard

1, 2, 3, 11 at midside nodes of modified displacement and temperature elements in Abaqus/Standard

Additional Solution Variables

The constant pressure hybrid element has one additional variable relating to pressure, and the linear pressure hybrid elements have four additional variables relating to pressure.

Element type C3D8IT has thirteen additional variables relating to the incompatible modes, and element type C3D8IHT has nine additional variables relating to the incompatible modes.

Element types C3D10MT and C3D10MHT have three additional displacement variables and one additional temperature variable.

Coupled Thermal-Electrical-Structural Elements

Q3D4$^{(S)}$	4-node linear displacement, electric potential and temperature
Q3D6$^{(S)}$	6-node linear displacement, electric potential and temperature
Q3D8$^{(S)}$	8-node trilinear displacement, electric potential and temperature
Q3D8H$^{(S)}$	8-node trilinear displacement, electric potential and temperature, hybrid with constant pressure
Q3D8R$^{(S)}$	8-node trilinear displacement, electric potential and temperature, reduced integration with hourglass control
Q3D8RH$^{(S)}$	8-node trilinear displacement, electric potential and temperature, reduced integration with hourglass control, hybrid with constant pressure
Q3D10M$^{(S)}$	10-node modified displacement, electric potential and temperature tetrahedron, with hourglass control
Q3D10MH$^{(S)}$	10-node modified displacement, electric potential and temperature tetrahedron, with hourglass control, hybrid with linear pressure
Q3D20$^{(S)}$	20-node triquadratic displacement, trilinear electric potential and trilinear temperature
Q3D20H$^{(S)}$	20-node triquadratic displacement, trilinear electric potential, trilinear temperature, hybrid with linear pressure
Q3D20R$^{(S)}$	20-node triquadratic displacement, trilinear electric potential, trilinear temperature, reduced integration
Q3D20RH$^{(S)}$	20-node triquadratic displacement, trilinear electric potential, trilinear temperature, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2, 3, 9, 11 at corner nodes

1, 2, 3 at midside nodes of second-order elements in Abaqus/Standard

1, 2, 3, 9, 11 at midside nodes of modified displacement and temperature elements in Abaqus/Standard

Additional Solution Variables

The constant pressure hybrid element has one additional variable relating to pressure, and the linear pressure hybrid elements have four additional variables relating to pressure.

Element types Q3D10M and Q3D10MH have three additional displacement variables, one additional electric potential variable, and one additional temperature variable.

Diffusive Heat Transfer or Mass Diffusion Elements

DC3D4$^{(S)}$	4-node linear tetrahedron
DC3D5$^{(S)}$	5-node linear pyramid
DC3D6$^{(S)}$	6-node linear wedge
DC3D8$^{(S)}$	8-node linear brick
DC3D8R$^{(S)}$	8-node linear brick, reduced integration, hourglass control
DC3D10$^{(S)}$	10-node quadratic tetrahedron
DC3D15$^{(S)}$	15-node quadratic wedge
DC3D20$^{(S)}$	20-node quadratic brick

Active Degrees of Freedom

11

Additional Solution Variables

None.

Forced Convection/Diffusion Elements

DCC3D8$^{(S)}$	8-node
DCC3D8D$^{(S)}$	8-node with dispersion control

Active Degrees of Freedom

11

Additional Solution Variables

None.

Coupled Thermal-Electrical Elements

DC3D4E$^{(S)}$	4-node linear tetrahedron
DC3D6E$^{(S)}$	6-node linear wedge
DC3D8E$^{(S)}$	8-node linear brick
DC3D10E$^{(S)}$	10-node quadratic tetrahedron
DC3D15E$^{(S)}$	15-node quadratic wedge
DC3D20E$^{(S)}$	20-node quadratic brick

Active Degrees of Freedom

9, 11

Additional Solution Variables

None.

Pore Pressure Elements

C3D4P$^{(S)}$	4-node linear displacement and pore pressure
C3D4PH$^{(S)}$	4-node linear displacement and pore pressure, hybrid with linear pressure
C3D6P$^{(S)}$	6-node linear displacement and pore pressure
C3D6PH$^{(S)}$	6-node linear displacement and pore pressure, hybrid with constant pressure
C3D8P$^{(S)}$	8-node trilinear displacement and pore pressure
C3D8PH$^{(S)}$	8-node trilinear displacement and pore pressure, hybrid with constant pressure
C3D8RP$^{(S)}$	8-node trilinear displacement and pore pressure, reduced integration
C3D8RPH$^{(S)}$	8-node trilinear displacement and pore pressure, reduced integration, hybrid with constant pressure
C3D10P$^{(S)}$	10-node triquadratic displacement, trilinear pore pressure
C3D10PH$^{(S)}$	10-node triquadratic displacement, trilinear pore pressure, hybrid with constant pressure
C3D10MP$^{(S)}$	10-node modified displacement and pore pressure tetrahedron, with hourglass control
C3D10MPH$^{(S)}$	10-node modified displacement and pore pressure tetrahedron, with hourglass control, hybrid with linear pressure
C3D20P$^{(S)}$	20-node triquadratic displacement, trilinear pore pressure
C3D20PH$^{(S)}$	20-node triquadratic displacement, trilinear pore pressure, hybrid with linear pressure
C3D20RP$^{(S)}$	20-node triquadratic displacement, trilinear pore pressure, reduced integration

C3D20RPH[(S)]	20-node triquadratic displacement, trilinear pore pressure, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2, 3 at midside nodes for all elements except C3D10MP and C3D10MPH, which also have degree of freedom 8 active at midside nodes

1, 2, 3, 8 at corner nodes

Additional Solution Variables

The constant pressure hybrid elements have one additional variable relating to the effective pressure stress, and the linear pressure hybrid elements have four additional variables relating to the effective pressure stress to permit fully incompressible material modeling.

Element types C3D10MP and C3D10MPH have three additional displacement variables and one additional pore pressure variable.

Coupled Temperature–Pore Pressure Elements

C3D4PT[(S)]	4-node trilinear displacement, pore pressure, and temperature
C3D4PHT[(S)]	4-node trilinear displacement, pore pressure, and temperature; hybrid with linear pressure
C3D6PT[(S)]	6-node trilinear displacement, pore pressure, and temperature
C3D6PHT[(S)]	6-node trilinear displacement, pore pressure, and temperature; hybrid with constant pressure
C3D8PT[(S)]	8-node trilinear displacement, pore pressure, and temperature
C3D8PHT[(S)]	8-node trilinear displacement, pore pressure, and temperature; hybrid with constant pressure
C3D8RPT[(S)]	8-node trilinear displacement, pore pressure, and temperature; reduced integration
C3D8RPHT[(S)]	8-node trilinear displacement, pore pressure, and temperature; reduced integration, hybrid with constant pressure
C3D10MPT[(S)]	10-node modified displacement, pore pressure, and temperature tetrahedron, with hourglass control
C3D10PT[(S)]	10-node triquadratic displacement, trilinear pore pressure, and temperature
C3D10PHT[(S)]	10-node triquadratic displacement, trilinear pore pressure, and temperature; hybrid with constant pressure

Active Degrees of Freedom

1, 2, 3, 8, 11

Additional Solution Variables

The constant pressure hybrid elements have one additional variable relating to the effective pressure stress to permit fully incompressible material modeling.

Element type C3D10MPT has three additional displacement variables, one additional pore pressure variable, and one additional temperature variable.

Acoustic Elements

AC3D4	4-node linear tetrahedron
AC3D5	5-node linear pyramid
AC3D6	6-node linear wedge
AC3D8$^{(S)}$	8-node linear brick
AC3D8R$^{(E)}$	8-node linear brick, reduced integration with hourglass control
AC3D10$^{(S)}$	10-node quadratic tetrahedron
AC3D15$^{(S)}$	15-node quadratic wedge
AC3D20$^{(S)}$	20-node quadratic brick

Active Degrees of Freedom

8

Additional Solution Variables

None.

Poroelastic Acoustic Elements

C3D4A$^{(S)}$	4-node linear tetrahedron
C3D6A$^{(S)}$	6-node linear wedge
C3D8A$^{(S)}$	8-node linear brick

Active Degrees of Freedom

1, 2, 3, 8

Additional Solution Variables

None.

Piezoelectric Elements

C3D4E$^{(S)}$	4-node linear tetrahedron
C3D6E$^{(S)}$	6-node linear wedge
C3D8E$^{(S)}$	8-node linear brick
C3D10E$^{(S)}$	10-node quadratic tetrahedron
C3D15E$^{(S)}$	15-node quadratic wedge
C3D20E$^{(S)}$	20-node quadratic brick
C3D20RE$^{(S)}$	20-node quadratic brick, reduced integration

Active Degrees of Freedom

1, 2, 3, 9

Additional Solution Variables

None.

Electromagnetic Elements

EMC3D4$^{(S)}$	4-node zero-order
EMC3D6$^{(S)}$	6-node zero-order
EMC3D8$^{(S)}$	8-node zero-order

Active Degrees of Freedom

Magnetic vector potential (for more information, see *Boundary Conditions* and *Boundary Conditions*).

Additional Solution Variables

None.

Coupled Thermal-Electrochemical Elements

QEC3D4$^{(S)}$	4-node linear electric potential in solid, temperature, electric potential in electrolyte, ion concentration, species concentration
QEC3D6$^{(S)}$	6-node linear electric potential in solid, temperature, electric potential in electrolyte, ion concentration, species concentration
QEC3D8$^{(S)}$	8-node trilinear electric potential in solid, temperature, electric potential in electrolyte, ion concentration, species concentration

QEC3D8R$^{(S)}$ 8-node trilinear electric potential in solid, temperature, electric potential in electrolyte, ion concentration, species concentration; reduced integration with hourglass control

Active Degrees of Freedom

9, 11, 32, 33, 34 at corner nodes

Additional Solution Variables

None.

Coupled Thermal-Electrochemical-Structural Elements

QEC3D4$^{(S)}$ 4-node linear displacement, electric potential in solid, temperature, electric potential in electrolyte, ion concentration, species concentration

QEC3D6$^{(S)}$ 6-node linear displacement, electric potential in solid, temperature, electric potential in electrolyte, ion concentration, species concentration

QEC3D8$^{(S)}$ 8-node trilinear displacement, electric potential in solid, temperature, electric potential in electrolyte, ion concentration, species concentration

QEC3D8R$^{(S)}$ 8-node trilinear displacement, electric potential in solid, temperature, electric potential in electrolyte, ion concentration, species concentration; reduced integration with hourglass control

Active Degrees of Freedom

1, 2, 3, 9, 11, 32, 33, 34 at corner nodes

Additional Solution Variables

None.

Coupled Thermal-Electrochemical-Structural–Pore Pressure Elements

QEC3D4$^{(S)}$ 4-node linear displacement, electric potential in solid, temperature, electric potential in electrolyte, and ion concentration, and either pore pressure or species concentration

QEC3D6$^{(S)}$ 6-node linear displacement, electric potential in solid, temperature, electric potential in electrolyte, and ion concentration, and either pore pressure or species concentration

QEC3D8$^{(S)}$ 8-node trilinear displacement, electric potential in solid, temperature, electric potential in electrolyte, and ion concentration, and either pore pressure or species concentration

QEC3D8R$^{(S)}$ 8-node trilinear displacement, electric potential in solid, temperature, electric potential in electrolyte, ion concentration, and either pore pressure or species concentration, and reduced integration with hourglass control

Active Degrees of Freedom

1, 2, 3, 9, 11, 32, 33, and either 8 or 34 at corner nodes

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage: **SOLID SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Solid** as the section **Category** and **Homogeneous** or **Electromagnetic, Solid** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are available for all elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *X*-direction.

Load ID (*DLOAD): BY
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *Y*-direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *Z*-direction.

Load ID (*DLOAD): BXNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *X*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BYNU
Abaqus/CAE Load/Interaction: Body force

Units: FL^{-3}

Description: Nonuniform body force in global *Y*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}

Description: Nonuniform body force in global *Z*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): CENT
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}(ML^{-3}T^{-2})$

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume, ω is the angular velocity). Not available for pore pressure elements.

Load ID (*DLOAD): CENTRIF
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): CORIO
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-4}T$ $(ML^{-3}T^{-1})$

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit volume, ω is the angular velocity). Not available for pore pressure elements.

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}

Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): HP*n* (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Hydrostatic pressure on face *n*, linear in global *Z*.

Load ID (*DLOAD): P*n*
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}

Description: Pressure on face *n*.

Load ID (*DLOAD): P*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Nonuniform pressure on face *n* with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): ROTA
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Rotary acceleration load (magnitude is input as α, where α is the rotary acceleration).

Load ID (*DLOAD): ROTDYNF
Abaqus/CAE Load/Interaction: Not supported
Units: T^{-1}
Description: Rotordynamic load (magnitude is input as ω, where ω is the angular velocity).

Load ID (*DLOAD): SBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-5}T^2$
Description: Stagnation body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): SP*n* (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T^2$
Description: Stagnation pressure on face *n*.

Load ID (*DLOAD): TRSHR*n*
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on face *n*.

Load ID (*DLOAD): TRSHR*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform shear traction on face *n* with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC*n*
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on face *n*.

Load ID (*DLOAD): TRVEC*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform general traction on face *n* with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): VBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T$
Description: Viscous body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): VP*n* (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-3}T$
Description: Viscous pressure on face *n*, applying a pressure proportional to the velocity normal to the face and opposing the motion.

Foundations

Foundations are available for Abaqus/Standard elements with displacement degrees of freedom. They are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): F*n* (S)
Abaqus/CAE Load/Interaction: Elastic foundation
Units: FL^{-3}
Description: Elastic foundation on face *n*.

Distributed Heat Fluxes

Distributed heat fluxes are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DFLUX): BF
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Heat body flux per unit volume.

Load ID (*DFLUX): BFNU
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Nonuniform heat body flux per unit volume with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard and *VDFLUX* in Abaqus/Explicit.

Load ID (*DFLUX): MBFNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: JT^{-1}
Description: Nonuniform moving or stationary concentrated heat fluxes with magnitudes supplied via user subroutine *UMDFLUX*.

Load ID (*DFLUX): S*n*
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Heat surface flux per unit area into face *n*.

Load ID (*DFLUX): S*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}$
Description: Nonuniform heat surface flux per unit area into face *n* with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard and *VDFLUX* in Abaqus/Explicit.

Film Conditions

Film conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*FILM): F*n*
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\theta^{-1}$
Description: Film coefficient and sink temperature (units of θ) provided on face *n*.

Load ID (*FILM): F*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of θ) provided on face n with magnitude supplied via user subroutine *FILM*.

Load ID (*FILM): FFS
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Film coefficient and sink temperature (units of θ) provided on all free faces of an element.

Load ID (*FILM): FFSNU
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of θ) provided on all free faces of an element with magnitude supplied in user subroutine.

Radiation Types

Radiation conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*RADIATE): Rn
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) provided on face n.

Load ID (*RADIATE): RFS
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) provided on free faces of an element.

Distributed Flows

Distributed flows are available for all elements with pore pressure degrees of freedom. They are specified as described in *Pore Fluid Flow*.

Load ID (*FLOW): Qn (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^3T^{-1}$
Description: Seepage coefficient and reference sink pore pressure (units of FL^{-2}) provided on face n.

Load ID (*FLOW): QnD (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^3T^{-1}$
Description: Drainage-only seepage coefficient provided on face n.

Load ID (*FLOW): QnNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^3T^{-1}$
Description: Nonuniform seepage coefficient and reference sink pore pressure (units of FL^{-2}) provided on face n with magnitude supplied via user subroutine *FLOW*.

Load ID (*DFLOW): Sn (S)
Abaqus/CAE Load/Interaction: Surface pore fluid

Units: LT^{-1}

Description: Prescribed pore fluid effective velocity (outward from the face) on face *n*.

Load ID (*DFLOW): S*n*NU (S)

Abaqus/CAE Load/Interaction: Not supported

Units: LT^{-1}

Description: Nonuniform prescribed pore fluid effective velocity (outward from the face) on face *n* with magnitude supplied via user subroutine *DFLOW*.

Distributed Impedances

Distributed impedances are available for all elements with acoustic pressure degrees of freedom. They are specified as described in *Acoustic and Shock Loads*.

Load ID (*IMPEDANCE): I*n*

Abaqus/CAE Load/Interaction: Not supported

Units: None

Description: Name of the impedance property that defines the impedance on face *n*.

Electric Fluxes

Electric fluxes are available for piezoelectric elements. They are specified as described in *Piezoelectric Analysis*.

Load ID (*DECHARGE): EBF (S)

Abaqus/CAE Load/Interaction: Body charge

Units: CL^{-3}

Description: Body flux per unit volume.

Load ID (*DECHARGE): ES*n* (S)

Abaqus/CAE Load/Interaction: Surface charge

Units: CL^{-2}

Description: Prescribed surface charge on face *n*.

Distributed Electric Current Densities

Distributed electric current densities are available for coupled thermal-electrical, coupled thermal-electrochemical, coupled thermal-electrical-structural, coupled thermal-electrochemical-structural, and electromagnetic elements. They are specified as described in *Coupled Thermal-Electrical Analysis*, *Coupled Thermal-Electrochemical Analysis*, *Fully Coupled Thermal-Electrical-Structural Analysis*, and *Eddy Current Analysis*.

Distributed fluid electric current densities are available for coupled thermal-electrochemical and coupled thermal-electrochemical-structural elements. They are specified as described in *Coupled Thermal-Electrochemical Analysis*.

Load ID (*DECURRENT): CBF (S)

Abaqus/CAE Load/Interaction: Body current

Units: $CL^{-3}T^{-1}$

Description: Volumetric current source density.

Load ID (*DECURRENT): CBFNU

Units: $CL^{-3}T^{-1}$

Description: Nonuniform volumetric current source density with magnitude supplied via user subroutine *DECURRENT*.

Load ID (*DECURRENT): CS*n* (S)
Abaqus/CAE Load/Interaction: Surface current
Units: $CL^{-2}T^{-1}$
Description: Current density on face *n*.

Load ID (*DECURRENT): CS*n*NU (S)
Units: $CL^{-2}T^{-1}$
Description: Nonuniform current density on face *n* with magnitude supplied via user subroutine *DECURRENT*.

Load ID (*DECURRENT): CJ (S)
Abaqus/CAE Load/Interaction: Body current density
Units: $CL^{-2}T^{-1}$
Description: Volume current density vector in an eddy current analysis.

Load ID (*DECURRENT): ECS*n* (S)
Abaqus/CAE Load/Interaction: Fluid surface current
Units: $CL^{-2}T^{-1}$
Description: Fluid current density on face *n*.

Distributed Concentration Fluxes

Distributed concentration fluxes are available for mass diffusion elements, coupled thermal-electrochemical and coupled thermal-electrochemical-structural elements. They are specified as described in *Mass Diffusion Analysis* and *Coupled Thermal-Electrochemical Analysis*.

Load ID (*DFLUX): BF (S)
Abaqus/CAE Load/Interaction: Body concentration flux
Units: PT^{-1}
Description: Concentration body flux per unit volume.

Load ID (*DFLUX): BFCE (S)
Abaqus/CAE Load/Interaction: Body concentration flux
Units: Mole $L^{-3}T^{-1}$
Description: Ion concentration body flux per unit volume.

Load ID (*DFLUX): BFNU (S)
Abaqus/CAE Load/Interaction: Body concentration flux
Units: PT^{-1}
Description: Nonuniform concentration body flux per unit volume with magnitude supplied via user subroutine *DFLUX*.

Load ID (*DFLUX): S*n* (S)
Abaqus/CAE Load/Interaction: Surface concentration flux
Units: PLT^{-1}
Description: Concentration surface flux per unit area into face *n*.

Load ID (*DFLUX): S*n*NU (S)
Abaqus/CAE Load/Interaction: Surface concentration flux
Units: PLT^{-1}
Description: Nonuniform concentration surface flux per unit area into face *n* with magnitude supplied via user subroutine *DFLUX*.

Multiphysics Loads

Multiphysics loads are available for elements used in the coupled thermal-electrochemical, coupled thermal-electrochemical-structural, or coupled thermal-electrochemical-structural–pore pressure procedures. They are specified as described in *Modeling Solid Electrodes in Lithium Metal Batteries*.

Load ID (*MULTIPHYSICS OPTION): QS$n^{(S)}$
Units: Units vary depending on the field.
Surface-based Butler-Volmer interaction on face *n*.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are available for all elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): HP (S)
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Hydrostatic pressure on the element surface, linear in global *Z*.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element surface.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Nonuniform pressure on the element surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DSLOAD): SP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-4}T^2$
Description: Stagnation pressure on the element surface.

Load ID (*DSLOAD): ST
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Surface tension on the element surface.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element surface.

Load ID (*DSLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Nonuniform shear traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element surface.

Load ID (*DSLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform general traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): VP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-3}T$
Description: Viscous pressure applied on the element surface. The viscous pressure is proportional to the velocity normal to the element face and opposing the motion.

Distributed Heat Fluxes

Surface-based heat fluxes are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DSFLUX): S
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Heat surface flux per unit area into the element surface.

Load ID (*DSFLUX): SNU
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Nonuniform heat surface flux per unit area into the element surface with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard and *VDFLUX* in Abaqus/Explicit.

Film Conditions

Surface-based film conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*SFILM): F
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\boldsymbol{\theta}^{-1}$

Description: Film coefficient and sink temperature (units of $\boldsymbol{\theta}$) provided on the element surface.

Load ID (*SFILM): FNU (S)
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\boldsymbol{\theta}^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of $\boldsymbol{\theta}$) provided on the element surface with magnitude supplied via user subroutine *FILM*.

Radiation Types

Surface-based radiation conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*SRADIATE): R
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) provided on the element surface.

Load ID (*SRADIATE): AVG
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless
Description: Emissivity provided on the element surface.

Distributed Flows

Surface-based flows are available for all elements with pore pressure degrees of freedom. They are specified as described in *Pore Fluid Flow*.

Load ID (*SFLOW): Q (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^3T^{-1}$

Description: Seepage coefficient and reference sink pore pressure (units of FL^{-2}) provided on the element surface.

Load ID (*SFLOW): QD (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^3T^{-1}$

Description: Drainage-only seepage coefficient provided on the element surface.

Load ID (*SFLOW): QNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^3T^{-1}$

Description: Nonuniform seepage coefficient and reference sink pore pressure (units of FL^{-2}) provided on the element surface with magnitude supplied via user subroutine *FLOW*.

Load ID (*DSFLOW): S (S)
Abaqus/CAE Load/Interaction: Surface pore fluid
Units: LT^{-1}

Description: Prescribed pore fluid effective velocity outward from the element surface.

Load ID (*DSFLOW): SNU (S)
Abaqus/CAE Load/Interaction: Surface pore fluid
Units: LT^{-1}

Description: Nonuniform prescribed pore fluid effective velocity outward from the element surface with magnitude supplied via user subroutine *DFLOW*.

Distributed Impedances

Surface-based impedances are available for all elements with acoustic pressure degrees of freedom. They are specified as described in *Acoustic and Shock Loads*.

Incident Wave Loading

Surface-based incident wave loads are available for all elements with displacement degrees of freedom or acoustic pressure degrees of freedom. They are specified as described in *Acoustic and Shock Loads*. If the incident wave field includes a reflection off a plane outside the boundaries of the mesh, this effect can be included.

Electric Fluxes

Surface-based electric fluxes are available for piezoelectric elements. They are specified as described in *Piezoelectric Analysis*.

Load ID (*DSECHARGE): ES (S)
Abaqus/CAE Load/Interaction: Surface charge
Units: CL^{-2}
Description: Prescribed surface charge on the element surface.

Distributed Electric Current Densities

Surface-based electric current densities are available for coupled thermal-electrical, coupled thermal-electrochemical, coupled thermal-electrical-structural, coupled thermal-electrochemical-structural, and electromagnetic elements. They are specified as described in *Coupled Thermal-Electrical Analysis*, *Coupled Thermal-Electrochemical Analysis*, *Fully Coupled Thermal-Electrical-Structural Analysis*, and *Eddy Current Analysis*.

Surface-based fluid electric current densities are available for coupled thermal-electrochemical and coupled thermal-electrochemical-structural elements. They are specified as described in *Coupled Thermal-Electrochemical Analysis*.

Load ID (*DSECURRENT): CS (S)
Abaqus/CAE Load/Interaction: Surface current
Units: $CL^{-2}T^{-1}$
Description: Current density on the element surface.

Load ID (*DSECURRENT): CSNU (S)
Units: $CL^{-2}T^{-1}$
Description: Nonuniform current density on the element surface with magnitude supplied via user subroutine *DECURRENT*.

Load ID (*DSECURRENT): CK (S)
Abaqus/CAE Load/Interaction: Surface current density
Units: $CL^{-1}T^{-1}$
Description: Surface current density vector in an eddy current analysis.

Load ID (*DSECURRENT): ECS (S)
Abaqus/CAE Load/Interaction: Fluid surface current
Units: $CL^{-2}T^{-1}$
Description: Fluid current density on the element surface.

Multiphysics Loads

Multiphysics loads are available for elements used in the coupled thermal-electrochemical, coupled thermal-electrochemical-structural, or coupled thermal-electrochemical-structural–pore pressure procedures. They are specified as described in *Modeling Solid Electrodes in Lithium Metal Batteries*.

Load ID (*MULTIPHYSICS LOAD): QS$n^{(S)}$
Units: Units vary depending on the field.
Surface-based Butler-Volmer interaction on face n.

Element Output

For most elements output is in global directions unless a local coordinate system is assigned to the element through the section definition (*Orientations*) in which case output is in the local coordinate system (which rotates with the motion in large-displacement analysis). See *State storage* for details.

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11 — ***XX***, direct stress.

S22 — ***YY***, direct stress.

S33 — ***ZZ***, direct stress.

S12 — ***XY***, shear stress.

S13 — ***XZ***, shear stress.

S23 — ***YZ***, shear stress.



Note: The order shown above is not the same as that used in user subroutine *VUMAT*.

Heat Flux Components

Available for elements with temperature degrees of freedom.

HFL1 — Heat flux in the *X*-direction.

HFL2 — Heat flux in the *Y*-direction.

HFL3 — Heat flux in the *Z*-direction.

Pore Fluid Velocity Components

Available for elements with pore pressure degrees of freedom.

FLVEL1 — Pore fluid effective velocity in the *X*-direction.

FLVEL2 — Pore fluid effective velocity in the *Y*-direction.

FLVEL3 Pore fluid effective velocity in the *Z*-direction.

Mass Concentration Flux Components

Available for elements with normalized concentration degrees of freedom.

MFL1 Concentration flux in the *X*-direction.

MFL2 Concentration flux in the *Y*-direction.

MFL3 Concentration flux in the *Z*-direction.

Electrical Potential Gradient

Available for elements with electrical potential degrees of freedom.

EPG1 Electrical potential gradient in the *X*-direction.

EPG2 Electrical potential gradient in the *Y*-direction.

EPG3 Electrical potential gradient in the *Z*-direction.

Electrical Flux Components

Available for piezoelectric elements.

EFLX1 Electrical flux in the *X*-direction.

EFLX2 Electrical flux in the *Y*-direction.

EFLX3 Electrical flux in the *Z*-direction.

Electrical Current Density Components

Available for coupled thermal-electrical and coupled thermal-electrical-structural elements.

ECD1 Electrical current density in the *X*-direction.

ECD2 Electrical current density in the Y-direction.

ECD3 Electrical current density in the *Z*-direction.

Electrical Field Components

Available for electromagnetic elements in an eddy current analysis.

EME1 Electric field in the *X*-direction.

EME2 Electric field in the *Y*-direction.

EME3 Electric field in the *Z*-direction.

Magnetic Flux Density Components

Available for electromagnetic elements.

EMB1 Magnetic flux density in the *X*-direction.

EMB2 Magnetic flux density in the *Y*-direction.

EMB3 Magnetic flux density in the *Z*-direction.

Magnetic Field Components

Available for electromagnetic elements.

EMH1 Magnetic field in the *X*-direction.

EMH2 Magnetic field in the *Y*-direction.

EMH3 Magnetic field in the *Z*-direction.

Eddy Current Density Components in an Eddy Current Analysis

Available for electromagnetic elements in an eddy current analysis.

EMCD1 Eddy current density in the *X*-direction.

EMCD2 Eddy current density in the *Y*-direction.

EMCD3 Eddy current density in the *Z*-direction.

Applied Volume Current Density Components in an Eddy Current or Magnetostatic Analysis

Available for electromagnetic elements in an eddy current or magnetostatic analysis.

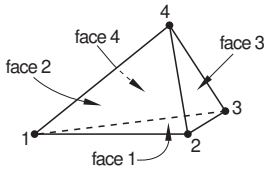
EMCDA1 Applied volume current density in the *X*-direction.

EMCDA2 Applied volume current density in the *Y*-direction.

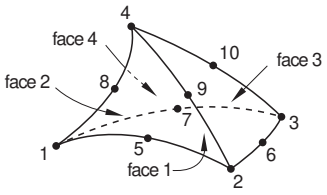
EMCDA3 Applied volume current density in the *Z*-direction.

Node Ordering and Face Numbering on Elements

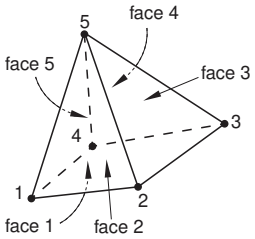
All Elements Except Variable Node Elements



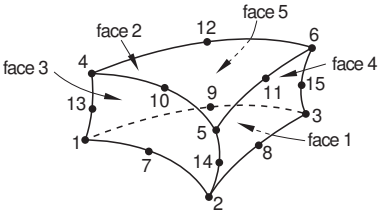
4 - node element



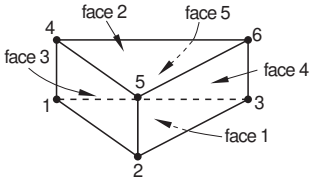
10 - node element



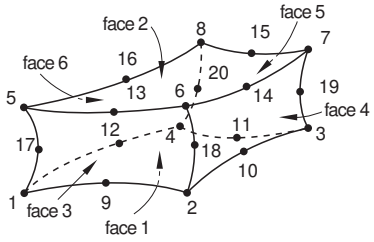
5 - node element



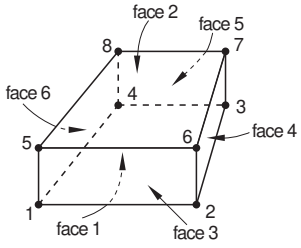
15 - node element



6 - node element



20 - node element



8 - node element

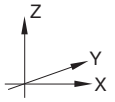


Table 1: Tetrahedral element faces

Face 1	1 – 2 – 3 face
Face 2	1 – 4 – 2 face
Face 3	2 – 4 – 3 face
Face 4	3 – 4 – 1 face

Table 2: Pyramid element faces

Face 1	1 – 2 – 3 – 4 face
Face 2	1 – 5 – 2 face
Face 3	2 – 5 – 3 face
Face 4	3 – 5 – 4 face
Face 5	4 – 5 – 1 face

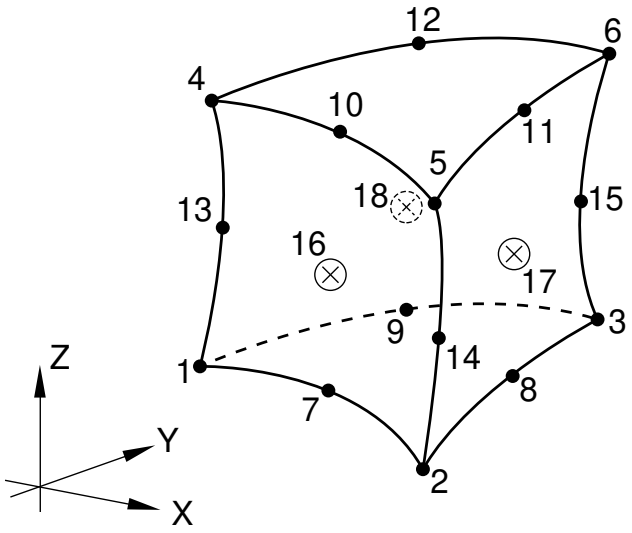
Table 3: Wedge (wedge) element faces

Face 1	1 – 2 – 3 face
Face 2	4 – 6 – 5 face
Face 3	1 – 4 – 5 – 2 face
Face 4	2 – 5 – 6 – 3 face
Face 5	3 – 6 – 4 – 1 face

Table 4: Hexahedron (brick) element faces

Face 1	1 – 2 – 3 – 4 face
Face 2	5 – 8 – 7 – 6 face
Face 3	1 – 5 – 6 – 2 face
Face 4	2 – 6 – 7 – 3 face
Face 5	3 – 7 – 8 – 4 face
Face 6	4 – 8 – 5 – 1 face

Variable Node Elements

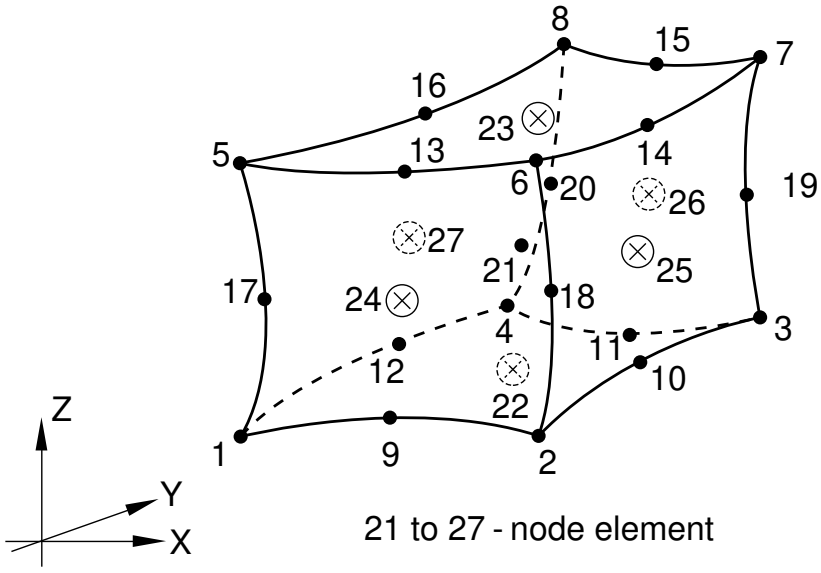


15 to 18 - node element

⊗ 16–18 are midface nodes on the three rectangular faces (see below for faces 1 to 5). These ⊗ nodes can be omitted from an element by entering a zero or blank in the corresponding position when giving the nodes on the element. Only nodes 16–18 can be omitted.

Table 5: Face location of nodes 16 to 18

Face node number	Corner nodes on face
16	1 – 4 – 5 – 2
17	2 – 5 – 6 – 3
18	3 – 6 – 4 – 1



21 to 27 - node element

Node 21 is located at the centroid of the element.

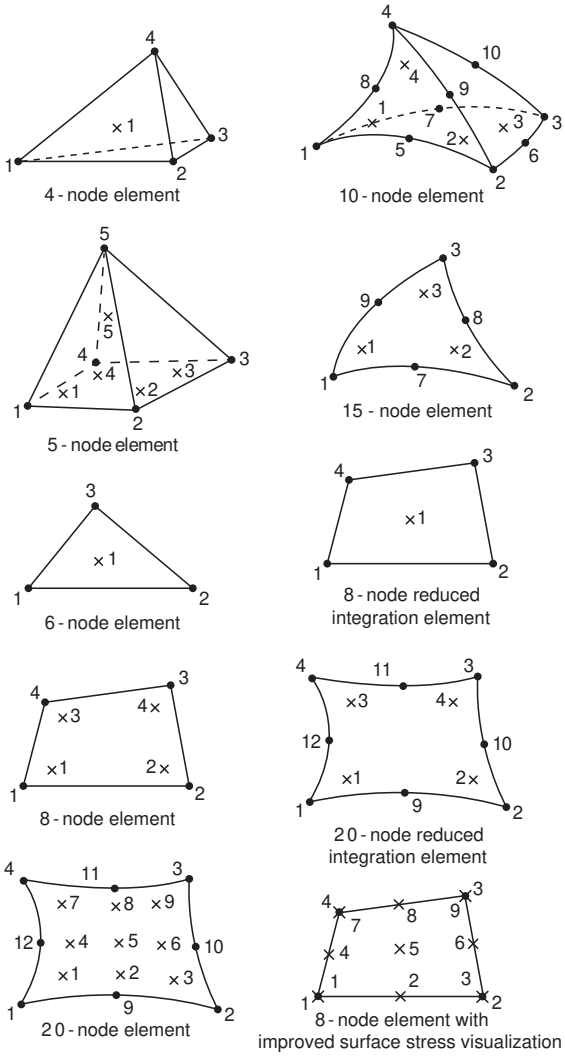
⊗ (nodes 22–27) are midface nodes on the six faces (see below for faces 1 to 6). These ⊗ nodes can be deleted from an element by entering a zero or blank in the corresponding position when giving the nodes on the element. Only nodes 22–27 can be omitted.

Table 6: Face location of nodes 22 to 27

Face node number	Corner nodes on face
22	1 – 2 – 3 – 4
23	5 – 8 – 7 – 6
24	1 – 5 – 6 – 2
25	2 – 6 – 7 – 3
26	3 – 7 – 8 – 4
27	4 – 8 – 5 – 1

Numbering of Integration Points for Output

All Elements Except Variable Node Elements



This shows the scheme in the layer closest to the 1–2–3 and 1–2–3–4 faces. The integration points in the second and third layers are numbered consecutively. Multiple layers are used for composite solid elements.

For heat transfer applications a different integration scheme is used for tetrahedral and wedge elements, as described in *Triangular, tetrahedral, and wedge elements*.

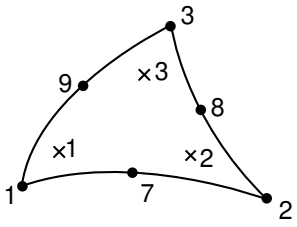
For linear wedges in Abaqus/Explicit reduced integration is used; therefore, a C3D6 element and a C3D6T element have only one integration point.

For the linear bricks C3D8S and C3D8HS in Abaqus/Standard improved stress visualization is obtained through a 27-point integration rule, consisting of 8 integration points at the elements' nodes, 12 integration points on the elements' edges, 6 integration points on the elements' sides, and one integration point inside the element.

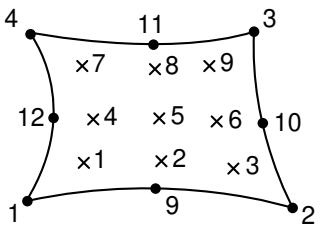
For the general-purpose C3D10HS 10-node tetrahedra in Abaqus/Standard improved stress visualization is obtained through an 11-point integration rule, consisting of 10 integration points at the elements' nodes and one integration point at their centroid.

For acoustic tetrahedra, pyramid, and wedges in Abaqus/Standard full integration is used; therefore, an AC3D4 element has 4 integration points, an AC3D5 element has 5 integration points, an AC3D6 element has 6 integration points, an AC3D10 element has 15 integration points, and an AC3D15 element has 18 integration points.

Variable Node Elements

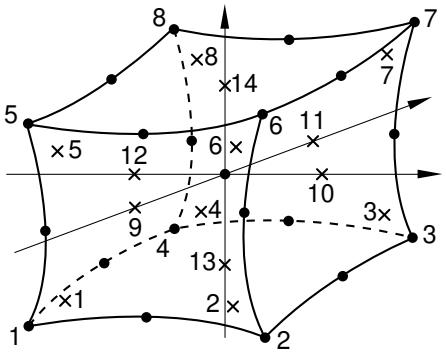


15 to 18 - node element



21 to 27 - node element

This shows the scheme in the layer closest to the 1–2–3 and 1–2–3–4 faces. The integration points in the second and third layers are numbered consecutively. Multiple layers are used for composite solid elements. The face nodes do not appear.



21 to 27 - node reduced integration element

Node 21 is located at the centroid of the element.

Cylindrical Solid Element Library

Products: Abaqus/Standard Abaqus/CAE

Overview

This section provides a reference to the cylindrical solid elements available in Abaqus/Standard.

References:

- *Solid (Continuum) Elements*
- **SOLID SECTION*

Element Types

CCL9	9-node cylindrical prism, linear interpolation in the radial plane and trigonometric interpolation along the circumferential direction
CCL9H	9-node cylindrical prism, linear interpolation in the radial plane and trigonometric interpolation along the circumferential direction, hybrid with constant pressure in plane and linear pressure in the circumferential direction
CCL12	12-node cylindrical brick, linear interpolation in the radial plane and trigonometric interpolation along the circumferential direction
CCL12H	12-node cylindrical brick, linear interpolation in the radial plane and trigonometric interpolation along the circumferential direction, hybrid with constant pressure in plane and linear pressure in circumferential direction
CCL18	18-node cylindrical prism, quadratic interpolation in the radial plane and trigonometric interpolation along the circumferential direction
CCL18H	18-node cylindrical prism, quadratic interpolation in the radial plane and trigonometric interpolation along the circumferential direction, hybrid with linear pressure in plane and linear pressure in the circumferential direction
CCL24	24-node cylindrical brick, quadratic interpolation in the radial plane and trigonometric interpolation along the circumferential direction
CCL24H	24-node cylindrical brick, quadratic interpolation in the radial plane and trigonometric interpolation along the circumferential direction, hybrid with linear pressure in plane and linear pressure in circumferential direction
CCL24R	24-node cylindrical brick, reduced integration, quadratic interpolation in the radial plane and trigonometric interpolation along the circumferential direction
CCL24RH	24-node cylindrical brick, reduced integration, quadratic interpolation in the radial plane and trigonometric interpolation along the circumferential direction, hybrid with linear pressure in plane and linear pressure in circumferential direction

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

The hybrid elements with constant pressure in plane have two additional variables relating to pressure, and the linear pressure hybrid elements have six additional variables relating to pressure.

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage:

**SOLID SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Solid** as the section **Category** and **Homogeneous** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Units: FL^{-3}
Description: Body force in global *X*-direction.

Load ID (*DLOAD): BY
Units: FL^{-3}
Description: Body force in global *Y*-direction.

Load ID (*DLOAD): BZ
Units: FL^{-3}
Description: Body force in global *Z*-direction.

Load ID (*DLOAD): BXNU
Units: FL^{-3}
Description: Nonuniform body force in global *X*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BYNU
Units: FL^{-3}

Description: Nonuniform body force in global *Y*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BZNU
Units: FL^{-3}
Description: Nonuniform body force in global *Z*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): CENT
Units: $FL^{-4}(ML^{-3}T^{-2})$

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): CENTRIF
Units: $FL^{-4}(ML^{-3}T^{-1})$

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): CORIO
Units: $FL^{-4}T$ $(ML^{-3}T^{-1})$

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): GRAV
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): HP*n*
Units: FL^{-2}
Description: Hydrostatic pressure on face *n*, linear in global *Z*.

Load ID (*DLOAD): P*n*
Units: FL^{-2}
Description: Pressure on face *n*.

Load ID (*DLOAD): ROTA
Units: T^{-2}
Description: Rotary acceleration load (magnitude is input as α, where α is the rotary acceleration).

Load ID (*DLOAD): ROTDYNF (S)
Units: T^{-1}
Description: Rotordynamic load (magnitude is input as ω, where ω is the angular velocity).

Load ID (*DLOAD): TRSHR*n*
Units: FL^{-2}
Description: Shear traction on face *n*.

Load ID (*DLOAD): TRSHR*n*NU (S)
Units: FL^{-2}
Description: Nonuniform shear traction on face *n* with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC*n*
Units: FL^{-2}
Description: General traction on face *n*.

Load ID (*DLOAD): TRVEC*n*NU (S)
Units: FL^{-2}
Description: Nonuniform general traction on face *n* with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Foundations

Foundations are available for all cylindrical elements. They are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): F*n*
Units: FL^{-3}
Description: Elastic foundation on face *n*.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are available for elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): HP
Units: FL^{-2}
Description: Hydrostatic pressure on the element surface, linear in global *Z*.

Load ID (*DSLOAD): P*n*
Units: FL^{-2}
Description: Pressure on the element surface.

Load ID (*DSLOAD): P*n*NU
Units: FL^{-2}
Description: Nonuniform pressure on the element surface with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DSLOAD): TRSHR
Units: FL^{-2}
Description: Shear traction on the element surface.

Load ID (*DSLOAD): TRSHRNU (S)
Units: FL^{-2}
Description: Nonuniform shear traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Units: FL^{-2}
Description: General traction on the element surface.

Load ID (*DSLOAD): TRVECNU (S)
Units: FL^{-2}
Description: Nonuniform general traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Element Output

Output is in a fixed cylindrical system (1=radial, 2=axial, 3=circumferential) unless a local coordinate system is assigned to the element through the section definition (*Orientations*) in which case output is in the local coordinate system (which rotates with the motion in large-displacement analysis). See *State storage* for details.

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11 Local 11 direct stress.

S22 Local 22 direct stress.

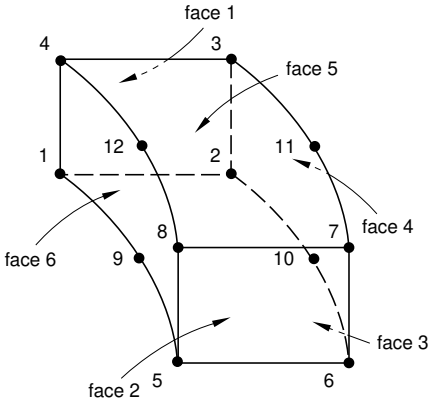
S33 Local 33 direct stress.

S12 Local 12 shear stress.

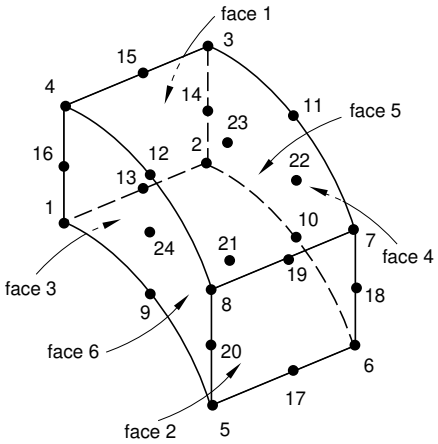
S13 Local 13 shear stress.

S23 Local 23 shear stress.

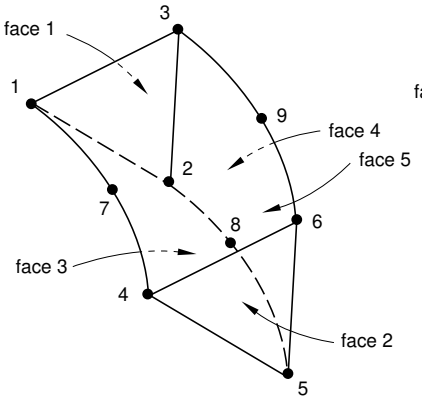
Node Ordering and Face Numbering on Elements



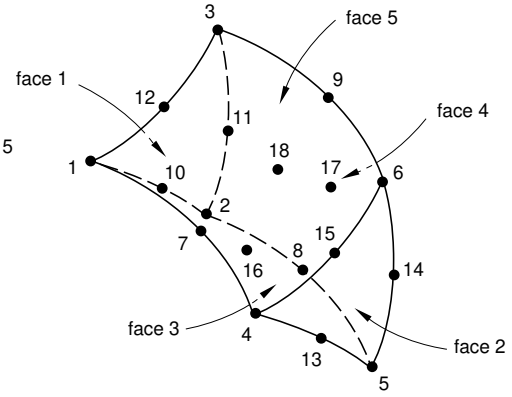
12-node element



24-node element



9-node element



18-node element

Table 1: 12-node and 24-node cylindrical element faces

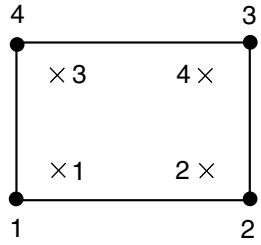
Face 1	1 – 2 – 3 – 4 face
Face 2	5 – 8 – 7 – 6 face
Face 3	1 – 5 – 6 – 2 face
Face 4	2 – 6 – 7 – 3 face
Face 5	3 – 7 – 8 – 4 face
Face 6	4 – 8 – 5 – 1 face

Table 2: 9-node and 18-node cylindrical element faces

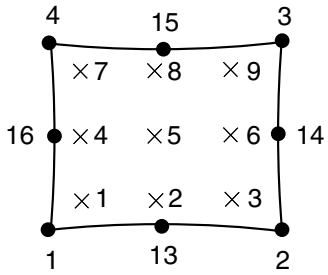
Face 1	1 – 2 – 3 face
Face 2	4 – 6 – 5 face
Face 3	1 – 4 – 5 – 2 face

Face 4	2 – 5 – 6 – 3 face
Face 5	3 – 6 – 4 – 1 face

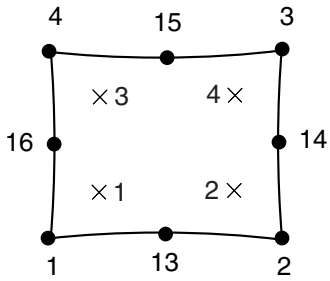
Numbering of Integration Points for Output



12-node element



24-node full integration element



24-node reduced integration element

This shows the scheme in the layer closest to the 1–2–3–4 face. The integration points in the second and third layers are numbered consecutively.

Axisymmetric Solid Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the axisymmetric solid elements available in Abaqus/Standard and Abaqus/Explicit.

Conventions

Coordinate 1 is r, coordinate 2 is z. At $\theta = 0$ the r-direction corresponds to the global x-direction and the z-direction corresponds to the global y-direction. This is important when data must be given in global directions. Coordinate 1 must be greater than or equal to zero.

Degree of freedom 1 is u_r, degree of freedom 2 is u_z. Generalized axisymmetric elements with twist have an additional degree of freedom, 5, corresponding to the twist angle ϕ (in radians).

Abaqus does not automatically apply any boundary conditions to nodes located along the symmetry axis. You must apply radial or symmetry boundary conditions on these nodes if desired.

In certain situations in Abaqus/Standard it might become necessary to apply radial boundary conditions on nodes that are located on the symmetry axis to obtain convergence in nonlinear problems. Therefore, the application of radial boundary conditions on nodes on the symmetry axis is recommended for nonlinear problems.

Point loads and moments, concentrated (nodal) fluxes, electrical currents, and seepage should be given as the value integrated around the circumference (that is, the total value on the ring).

References:

- *Solid (Continuum) Elements*
- **SOLID SECTION*

Element Types

Stress/Displacement Elements without Twist

CAX3	3-node linear
CAX3H$^{(S)}$	3-node linear, hybrid with constant pressure
CAX4$^{(S)}$	4-node bilinear
CAX4H$^{(S)}$	4-node bilinear, hybrid with constant pressure
CAX4I$^{(S)}$	4-node bilinear, incompatible modes
CAX4IH$^{(S)}$	4-node bilinear, incompatible modes, hybrid with linear pressure
CAX4R	4-node bilinear, reduced integration with hourglass control
CAX4RH$^{(S)}$	4-node bilinear, reduced integration with hourglass control, hybrid with constant pressure
CAX6$^{(S)}$	6-node quadratic

CAX6H$^{(S)}$	6-node quadratic, hybrid with linear pressure
CAX6M	6-node modified axisymmetric triangle, hourglass control
CAX6MH$^{(S)}$	6-node modified, with hourglass control, hybrid with linear pressure
CAX8$^{(S)}$	8-node biquadratic
CAX8H$^{(S)}$	8-node biquadratic, hybrid with linear pressure
CAX8R$^{(S)}$	8-node biquadratic, reduced integration
CAX8RH$^{(S)}$	8-node biquadratic, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2

Additional Solution Variables

The constant pressure hybrid elements have one additional variable and the linear pressure elements have three additional variables relating to pressure.

Element types CAX4I and CAX4IH have five additional variables relating to the incompatible modes.

Element types CAX6M and CAX6MH have two additional displacement variables.

Stress/Displacement Elements with Twist

CGAX3$^{(S)}$	3-node linear
CGAX3H$^{(S)}$	3-node linear, hybrid with constant pressure
CGAX4$^{(S)}$	4-node bilinear
CGAX4H$^{(S)}$	4-node bilinear, hybrid with constant pressure
CGAX4R$^{(S)}$	4-node bilinear, reduced integration with hourglass control
CGAX4RH$^{(S)}$	4-node bilinear, reduced integration with hourglass control, hybrid with constant pressure
CGAX6$^{(S)}$	6-node quadratic
CGAX6H$^{(S)}$	6-node quadratic, hybrid with linear pressure
CGAX6M$^{(S)}$	6-node modified, with hourglass control
CGAX6MH$^{(S)}$	6-node modified, with hourglass control, hybrid with linear pressure
CGAX8$^{(S)}$	8-node biquadratic
CGAX8H$^{(S)}$	8-node biquadratic, hybrid with linear pressure
CGAX8R$^{(S)}$	8-node biquadratic, reduced integration
CGAX8RH$^{(S)}$	8-node biquadratic, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2, 5

Additional Solution Variables

The constant pressure hybrid elements have one additional variable and the linear pressure elements have three additional variables relating to pressure.

Element types CGAX6M and CGAX6MH have three additional displacement variables.

Diffusive Heat Transfer or Mass Diffusion Elements

DCAX3$^{(S)}$	3-node linear
DCAX4$^{(S)}$	4-node linear
DCAX6$^{(S)}$	6-node quadratic
DCAX8$^{(S)}$	8-node quadratic

Active Degrees of Freedom

11

Additional Solution Variables

None.

Forced Convection/Diffusion Elements

DCCAX2$^{(S)}$	2-node
DCCAX2D$^{(S)}$	2-node with dispersion control
DCCAX4$^{(S)}$	4-node
DCCAX4D$^{(S)}$	4-node with dispersion control

Active Degrees of Freedom

11

Additional Solution Variables

None.

Coupled Thermal-Electrical Elements

DCAX3E$^{(S)}$	3-node linear
DCAX4E$^{(S)}$	4-node linear
DCAX6E$^{(S)}$	6-node quadratic
DCAX8E$^{(S)}$	8-node quadratic

Active Degrees of Freedom

9, 11

Additional Solution Variables

None.

Coupled Temperature-Displacement Elements without Twist

CAX3T	3-node axisymmetric thermally coupled triangle, linear displacement and temperature
CAX4T$^{(S)}$	4-node bilinear displacement and temperature
CAX4HT$^{(S)}$	4-node bilinear displacement and temperature, hybrid with constant pressure
CAX4RT	4-node bilinear displacement and temperature, reduced integration with hourglass control
CAX4RHT$^{(S)}$	4-node bilinear displacement and temperature, reduced integration with hourglass control, hybrid with constant pressure
CAX6MT	6-node modified axisymmetric thermally coupled triangle, hourglass control
CAX6MHT$^{(S)}$	6-node modified displacement and temperature, with hourglass control, hybrid with linear pressure
CAX8T$^{(S)}$	8-node biquadratic displacement, bilinear temperature
CAX8HT$^{(S)}$	8-node biquadratic displacement, bilinear temperature, hybrid with linear pressure
CAX8RT$^{(S)}$	8-node biquadratic displacement, bilinear temperature, reduced integration
CAX8RHT$^{(S)}$	8-node biquadratic displacement, bilinear temperature, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2, 11 at corner nodes

1, 2 at midside nodes of second-order elements in Abaqus/Standard

1, 2, 11 at midside nodes of the modified displacement and temperature elements in Abaqus/Standard

Additional Solution Variables

The constant pressure hybrid elements have one additional variable and the linear pressure elements have three additional variables relating to pressure.

Element types CAX6MT and CAX6MHT have two additional displacement variables and one additional temperature variable.

Coupled Temperature-Displacement Elements with Twist

CGAX3T$^{(S)}$	3-node linear displacement and temperature
CGAX3HT$^{(S)}$	3-node linear displacement and temperature, hybrid with constant pressure
CGAX4T$^{(S)}$	4-node bilinear displacement and temperature
CGAX4HT$^{(S)}$	4-node bilinear displacement and temperature, hybrid with constant pressure
CGAX4RT$^{(S)}$	4-node bilinear displacement and temperature, reduced integration with hourglass control
CGAX4RHT$^{(S)}$	4-node bilinear displacement and temperature, reduced integration with hourglass control, hybrid with constant pressure
CGAX6MT$^{(S)}$	6-node modified displacement and temperature, with hourglass control
CGAX6MHT$^{(S)}$	6-node modified displacement and temperature, with hourglass control, hybrid with constant pressure
CGAX8T$^{(S)}$	8-node biquadratic displacement, bilinear temperature
CGAX8HT$^{(S)}$	8-node biquadratic displacement, bilinear temperature, hybrid with linear pressure
CGAX8RT$^{(S)}$	8-node biquadratic displacement, bilinear temperature, reduced integration
CGAX8RHT$^{(S)}$	8-node biquadratic displacement, bilinear temperature, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2, 5, 11 at corner nodes

1, 2, 5 at midside nodes of second-order elements

1, 2, 5, 11 at midside nodes of the modified displacement and temperature elements

Additional Solution Variables

The constant pressure hybrid elements have one additional variable and the linear pressure elements have three additional variables relating to pressure.

Element types CGAX6MT and CGAX6MHT have two additional displacement variables and one additional temperature variable.

Pore Pressure Elements

CAX4P[(S)]	4-node bilinear displacement and pore pressure
CAX4PH[(S)]	4-node bilinear displacement and pore pressure, hybrid with constant pressure
CAX4RP[(S)]	4-node bilinear displacement and pore pressure, reduced integration with hourglass control
CAX4RPH[(S)]	4-node bilinear displacement and pore pressure, reduced integration with hourglass control, hybrid with constant pressure
CAX6MP[(S)]	6-node modified displacement and pore pressure, with hourglass control
CAX6MPH[(S)]	6-node modified displacement and pore pressure, with hourglass control, hybrid with linear pressure
CAX8P[(S)]	8-node biquadratic displacement, bilinear pore pressure
CAX8PH[(S)]	8-node biquadratic displacement, bilinear pore pressure, hybrid with linear pressure
CAX8RP[(S)]	8-node biquadratic displacement, bilinear pore pressure, reduced integration
CAX8RPH[(S)]	8-node biquadratic displacement, bilinear pore pressure, reduced integration, hybrid with linear pressure

Active Degrees of Freedom

1, 2, 8 at corner nodes

1, 2 at midside nodes

Additional Solution Variables

The constant pressure hybrid elements have one additional variable relating to the effective pressure stress, and the linear pressure hybrid elements have three additional variables relating to the effective pressure stress to permit fully incompressible material modeling.

Element types CAX6MP and CAX6MPH have two additional displacement variables and one additional pore pressure variable.

Coupled Temperature–Pore Pressure Elements

CAX4PT[(S)]	4-node bilinear displacement, pore pressure, and temperature
CAX4RPT[(S)]	4-node bilinear displacement, pore pressure, and temperature; reduced integration with hourglass control
CAX4RPHT[(S)]	4-node bilinear displacement, pore pressure, and temperature; reduced integration with hourglass control, hybrid with constant pressure

Active Degrees of Freedom

1, 2, 8, 11

Additional Solution Variables

The constant pressure hybrid elements have one additional variable relating to the effective pressure stress to permit fully incompressible material modeling.

Coupled Thermal-Electrical-Structural Elements

QAX3$^{(S)}$	3-node linear displacement, electric potential, temperature
QAX4$^{(S)}$	4-node bilinear displacement, electric potential, temperature
QAX6M$^{(S)}$	6-node modified displacement, electric potential, temperature; with hourglass control
QAX8$^{(S)}$	8-node biquadratic displacement, bilinear electric potential; bilinear temperature

Active Degrees of Freedom

1, 2, 9, 11 at corner nodes

1, 2 at midside nodes of second-order elements

1, 2, 9, 11 at midside nodes of the modified displacement, electric potential, and temperature elements

Additional Solution Variables

Element type QAX6M has two additional displacement variables, one additional electric potential variable, and one additional temperature variable.

Acoustic Elements

ACAX3	3-node linear
ACAX4R$^{(E)}$	4-node linear, reduced integration with hourglass control
ACAX4$^{(S)}$	4-node linear
ACAX6$^{(S)}$	6-node quadratic
ACAX8$^{(S)}$	8-node quadratic

Active Degrees of Freedom

8

Additional Solution Variables

None.

Piezoelectric Elements

CAX3E$^{(S)}$	3-node linear
CAX4E$^{(S)}$	4-node bilinear
CAX6E$^{(S)}$	6-node quadratic
CAX8E$^{(S)}$	8-node biquadratic
CAX8RE$^{(S)}$	8-node biquadratic, reduced integration

Active Degrees of Freedom

1, 2, 9

Additional Solution Variables

None.

Nodal Coordinates Required

r, z at $\theta = 0$

Element Property Definition

For element types DCCAX2 and DCCAX2D, you must specify the channel thickness of the element in the (r–z) plane. The default is unit thickness if no thickness is given.

For all other elements, you do not need to specify the thickness.

Input File Usage:

**SOLID SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Solid** as the section **Category** and **Homogeneous** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are available for all elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*. Distributed load magnitudes are per unit area or per unit volume. They do not need to be multiplied by 2π.

Load ID (*DLOAD): BR
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}

Description: Body force in radial direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in axial direction.

Load ID (*DLOAD): BRNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in radial direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in axial direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): CENT (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}M^{-3}T^{-2}$

Description: Centrifugal load (magnitude input as $\rho\omega^2$, where ρ is the mass density per unit volume, ω is the angular velocity). Not available for pore pressure elements.

Load ID (*DLOAD): CENTRIF (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): HP*n* (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Hydrostatic pressure on face *n*, linear in global *Y*.

Load ID (*DLOAD): P*n*
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on face *n*.

Load ID (*DLOAD): P*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure on face *n* with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): SBF (E)
Abaqus/CAE Load/Interaction: Not supported

Units: $FL^{-5}T^2$

Description: Stagnation body force in radial and axial directions.

Load ID (*DLOAD): SP*n* (E)

Abaqus/CAE Load/Interaction: Not supported

Units: $FL^{-4}T^2$

Description: Stagnation pressure on face *n*.

Load ID (*DLOAD): TRSHR*n*

Abaqus/CAE Load/Interaction: **Surface traction**

Units: FL^{-2}

Description: Shear traction on face *n*.

Load ID (*DLOAD): TRSHR*n*NU (S)

Abaqus/CAE Load/Interaction: Not supported

Units: FL^{-2}

Description: Nonuniform shear traction on face *n* with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC*n*

Abaqus/CAE Load/Interaction: **Surface traction**

Units: FL^{-2}

Description: General traction on face *n*.

Load ID (*DLOAD): TRVEC*n*NU (S)

Abaqus/CAE Load/Interaction: Not supported

Units: FL^{-2}

Description: Nonuniform general traction on face *n* with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): VBF (E)

Abaqus/CAE Load/Interaction: Not supported

Units: $FL^{-4}T$

Description: Viscous body force in radial and axial directions.

Load ID (*DLOAD): VP*n* (E)

Abaqus/CAE Load/Interaction: Not supported

Units: $FL^{-3}T$

Description: Viscous pressure on face *n*, applying a pressure proportional to the velocity normal to the face and opposing the motion.

Foundations

Foundations are available for Abaqus/Standard elements with displacement degrees of freedom. They are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): F*n* (S)

Abaqus/CAE Load/Interaction: **Elastic foundation**

Units: FL^{-3}

Description: Elastic foundation on face *n*. For CGAX elements the elastic foundations are applied to degrees of freedom u_r and u_z only.

Distributed Heat Fluxes

Distributed heat fluxes are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*. Distributed heat flux magnitudes are per unit area or per unit volume. They do not need to be multiplied by 2π.

Load ID (*DFLUX): BF
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Heat body flux per unit volume.

Load ID (*DFLUX): BFNU
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Nonuniform heat body flux per unit volume with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard and *VDFLUX* in Abaqus/Explicit.

Load ID (*DFLUX): S*n*
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Heat surface flux per unit area into face *n*.

Load ID (*DFLUX): S*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}$
Description: Nonuniform heat surface flux per unit area into face *n* with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard and *VDFLUX* in Abaqus/Explicit.

Film Conditions

Film conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*FILM): F*n*
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\theta^{-1}$
Description: Film coefficient and sink temperature (units of θ) provided on face *n*.

Load ID (*FILM): F*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\theta^{-1}$
Description: Nonuniform film coefficient and sink temperature (units of θ) provided on face *n* with magnitude supplied via user subroutine *FILM*.

Radiation Types

Radiation conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*RADIATE): R*n*
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature provided for face *n*.

Distributed Flows

Distributed flows are available for all elements with pore pressure degrees of freedom. They are specified as described in *Pore Fluid Flow*. Distributed flow magnitudes are per unit area or per unit volume. They do not need to be multiplied by $\mathbf{2\pi}$.

Load ID (*FLOW): Q*n* (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^3T^{-1}$
Description: Seepage coefficient and reference sink pore pressure (units of FL^{-2}) provided on face *n*.

Load ID (*FLOW): Q*n*D (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^3T^{-1}$
Description: Drainage-only seepage coefficient provided on face *n*.

Load ID (*FLOW): Q*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^3T^{-1}$
Description: Nonuniform seepage coefficient and reference sink pore pressure (units of FL^{-2}) provided on face *n* with magnitude supplied via user subroutine *FLOW*.

Load ID (*DFLOW): S*n* (S)
Abaqus/CAE Load/Interaction: Surface pore fluid
Units: LT^{-1}
Description: Prescribed pore fluid effective velocity (outward from the face) on face *n*.

Load ID (*DFLOW): S*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: LT^{-1}
Description: Nonuniform prescribed pore fluid effective velocity (outward from the face) on face *n* with magnitude supplied via user subroutine *DFLOW*.

Distributed Impedances

Distributed impedances are available for all elements with acoustic pressure degrees of freedom. They are specified as described in *Acoustic and Shock Loads*.

Load ID (*IMPEDANCE): I*n*
Abaqus/CAE Load/Interaction: Not supported
Units: None
Description: Name of the impedance property that defines the impedance on face *n*.

Electric Fluxes

Electric fluxes are available for piezoelectric elements. They are specified as described in *Piezoelectric Analysis*.

Load ID (*DECHARGE): EBF (S)
Abaqus/CAE Load/Interaction: Body charge
Units: CL^{-3}
Description: Body flux per unit volume.

Load ID (*DECHARGE): ES*n* (S)
Abaqus/CAE Load/Interaction: Surface charge
Units: CL^{-2}
Description: Prescribed surface charge on face *n*.

Distributed Electric Current Densities

Distributed electric current densities are available for coupled thermal-electrical elements. They are specified as described in *Coupled Thermal-Electrical Analysis*.

Load ID (*DECURRENT): CBF (S)
Abaqus/CAE Load/Interaction: Body current
Units: $CL^{-3}T^{-1}$
Description: Volumetric current source density.

Load ID (*DECURRENT): CS*n* (S)
Abaqus/CAE Load/Interaction: Surface current
Units: $CL^{-2}T^{-1}$
Description: Current density on face *n*.

Distributed Concentration Fluxes

Distributed concentration fluxes are available for mass diffusion elements. They are specified as described in *Mass Diffusion Analysis*.

Load ID (*DFLUX): BF (S)
Abaqus/CAE Load/Interaction: Body concentration flux
Units: PT^{-1}
Description: Concentration body flux per unit volume.

Load ID (*DFLUX): BFNU (S)
Abaqus/CAE Load/Interaction: Body concentration flux
Units: PT^{-1}
Description: Nonuniform concentration body flux per unit volume with magnitude supplied via user subroutine *DFLUX*.

Load ID (*DFLUX): S*n* (S)
Abaqus/CAE Load/Interaction: Surface concentration flux
Units: PLT^{-1}
Description: Concentration surface flux per unit area into face *n*.

Load ID (*DFLUX): S*n*NU (S)
Abaqus/CAE Load/Interaction: Surface concentration flux
Units: PLT^{-1}
Description: Nonuniform concentration surface flux per unit area into face *n* with magnitude supplied via user subroutine *DFLUX*.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are available for all elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*. Distributed load magnitudes are per unit area or per unit volume. They do not need to be multiplied by 2π.

Load ID (*DSLOAD): HP (S)
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Hydrostatic pressure on the element surface, linear in global *Y*.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element surface.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Nonuniform pressure on the element surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DSLOAD): SP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-4}T^2$
Description: Stagnation pressure on the element surface.

Load ID (*DSLOAD): ST
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Surface tension on the element surface.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element surface.

Load ID (*DSLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform shear traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element surface.

Load ID (*DSLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Surface traction

Units: FL^{-2}

Description: Nonuniform general traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): VP (E)

Abaqus/CAE Load/Interaction: Pressure

Units: $FL^{-3}T$

Description: Viscous pressure applied on the element surface. The viscous pressure is proportional to the velocity normal to the face and opposing the motion.

Distributed Heat Fluxes

Surface-based heat fluxes are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*. Distributed heat flux magnitudes are per unit area or per unit volume. They do not need to be multiplied by 2π.

Load ID (*DSFLUX): S

Abaqus/CAE Load/Interaction: Surface heat flux

Units: $JL^{-2}T^{-1}$

Description: Heat surface flux per unit area into the element surface.

Load ID (*DSFLUX): SNU

Abaqus/CAE Load/Interaction: Surface heat flux

Units: $JL^{-2}T^{-1}$

Description: Nonuniform heat surface flux per unit area into the element surface with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard and *VDFLUX* in Abaqus/Explicit.

Film Conditions

Surface-based film conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*SFILM): F

Abaqus/CAE Load/Interaction: Surface film condition

Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Film coefficient and sink temperature (units of θ) provided on the element surface.

Load ID (*SFILM): FNU (S)

Abaqus/CAE Load/Interaction: Surface film condition

Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of θ) provided on the element surface with magnitude supplied via user subroutine *FILM*.

Radiation Types

Surface-based radiation conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*SRADIATE): R

Abaqus/CAE Load/Interaction: Surface radiation

Units: Dimensionless

Description: Emissivity and sink temperature provided for the element surface.

Distributed Flows

Surface-based distributed flows are available for all elements with pore pressure degrees of freedom. They are specified as described in *Pore Fluid Flow*. Distributed flow magnitudes are per unit area or per unit volume. They do not need to be multiplied by 2π.

Load ID (*SFLOW): Q (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^{3}T^{-1}$
Description: Seepage coefficient and reference sink pore pressure (units of FL^{-2}) provided on the element surface.

Load ID (*SFLOW): QD (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^{3}T^{-1}$
Description: Drainage-only seepage coefficient provided on the element surface.

Load ID (*SFLOW): QNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $F^{-1}L^{3}T^{-1}$
Description: Nonuniform seepage coefficient and reference sink pore pressure (units of FL^{-2}) provided on the element surface with magnitude supplied via user subroutine *FLOW*.

Load ID (*DSFLOW): S (S)
Abaqus/CAE Load/Interaction: Surface pore fluid
Units: LT^{-1}
Description: Prescribed pore fluid effective velocity outward from the element surface.

Load ID (*DSFLOW): SNU (S)
Abaqus/CAE Load/Interaction: Surface pore fluid
Units: LT^{-1}
Description: Nonuniform prescribed pore fluid effective velocity outward from the element surface with magnitude supplied via user subroutine *DFLOW*.

Distributed Impedances

Surface-based impedances are available for all elements with acoustic pressure degrees of freedom. They are specified as described in *Acoustic and Shock Loads*.

Incident Wave Loading

Surface-based incident wave loads are available for all elements with displacement degrees of freedom or acoustic pressure degrees of freedom. They are specified as described in *Acoustic and Shock Loads*. If the incident wave field includes a reflection off a plane outside the boundaries of the mesh, this effect can be included.

Electric Fluxes

Surface-based electric fluxes are available for piezoelectric elements. They are specified as described in *Piezoelectric Analysis*.

Load ID (*DSECHARGE): ES (S)
Abaqus/CAE Load/Interaction: Surface charge
Units: CL^{-2}

Description: Prescribed surface charge on the element surface.

Distributed Electric Current Densities

Surface-based electric current densities are available for coupled thermal-electrical elements. They are specified as described in *Coupled Thermal-Electrical Analysis*.

Load ID (*DSECURRENT): CS (S)
Abaqus/CAE Load/Interaction: Surface current
Units: $CL^{-2}T^{-1}$
Description: Current density on the element surface.

Element Output

Output is in global directions unless a local coordinate system is assigned to the element through the section definition (*Orientations*) in which case output is in the local coordinate system (which rotates with the motion in large-displacement analysis). See *State storage* for details. For regular axisymmetric elements, the local orientation must be in the r–z plane with θ being a principal direction. For generalized axisymmetric elements with twist, the local orientation can be arbitrary.

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

For elements with displacement degrees of freedom without twist:

S11 Stress in the radial direction or in the local 1-direction.

S22 Stress in the axial direction or in the local 2-direction.

S33 Hoop direct stress.

S12 Shear stress.

For elements with displacement degrees of freedom with twist:

S11 Stress in the radial direction or in the local 1-direction.

S22 Stress in the axial direction or in the local 2-direction.

S33 Stress in the circumferential direction or in the local 3-direction.

S12 Shear stress.

S13 Shear stress.

S23 Shear stress.

Heat Flux Components

Available for elements with temperature degrees of freedom.

HFL1 Heat flux in the radial direction or in the local 1-direction.

HFL2 Heat flux in the axial direction or in the local 2-direction.

Pore Fluid Velocity Components

Available for elements with pore pressure degrees of freedom, except for acoustic elements.

FLVEL1 Pore fluid effective velocity in the radial direction or in the local 1-direction.

FLVEL2 Pore fluid effective velocity in the axial direction or in the local 2-direction.

Mass Concentration Flux Components

Available for elements with normalized concentration degrees of freedom.

MFL1 Concentration flux in the radial direction or in the local 1-direction.

MFL2 Concentration flux in the axial direction or in the local 2-direction.

Electrical Potential Gradient

Available for elements with electrical potential degrees of freedom.

EPG1 Electrical potential gradient in the 1-direction.

EPG2 Electrical potential gradient in the 2-direction.

Electrical Flux Components

Available for piezoelectric elements.

EFLX1 Electrical flux in the 1-direction.

EFLX2 Electrical flux in the 2-direction.

Electrical Current Density Components

Available for coupled thermal-electrical elements.

ECD1 Electrical current density in the 1-direction.

ECD2 Electrical current density in the 2-direction.

Node Ordering and Face Numbering on Elements

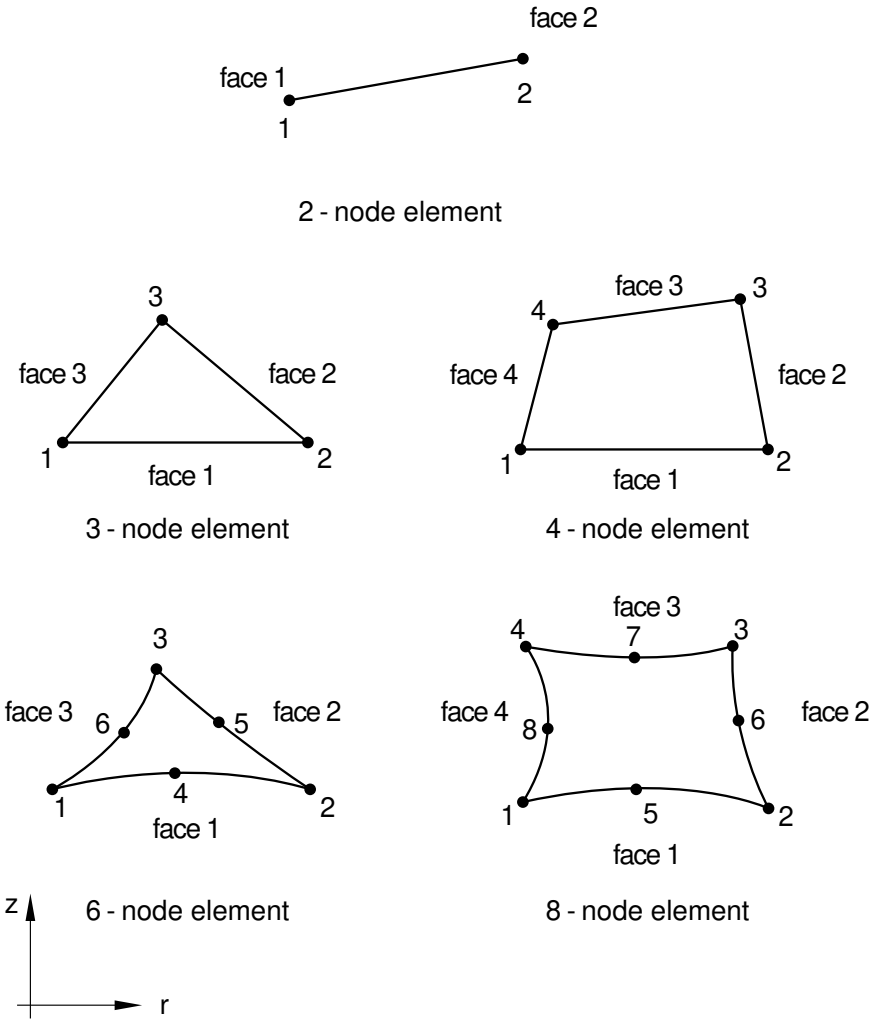


Table 1: 2-node element faces

Face 1	Section at node 1
Face 2	Section at node 2

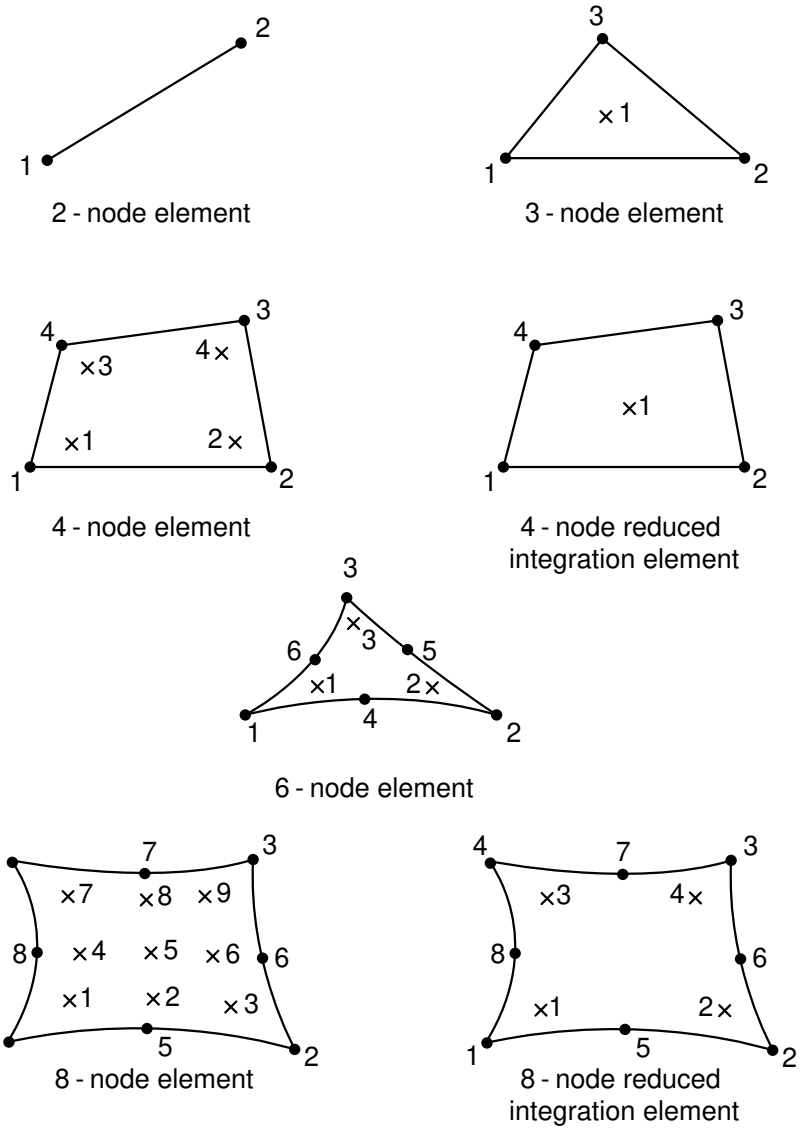
Table 2: Triangular element faces

Face 1	1 – 2 face
Face 2	2 – 3 face
Face 3	3 – 1 face

Table 3: Quadrilateral element faces

Face 1	1 – 2 face
Face 2	2 – 3 face
Face 3	3 – 4 face
Face 4	4 – 1 face

Numbering of Integration Points for Output



For heat transfer applications a different integration scheme is used for triangular elements, as described in *Triangular, tetrahedral, and wedge elements*.

Axisymmetric Solid Elements with Nonlinear, Asymmetric Deformation

Products: Abaqus/Standard Abaqus/CAE

Overview

This section provides a reference to the axisymmetric solid elements available in Abaqus/Standard. These elements are intended for analysis of hollow bodies, such as pipes and pressure vessels. They can also be used to model solid bodies, but spurious stresses may occur at zero radius, particularly if transverse shear loads are applied.

Conventions

Coordinate 1 is r, coordinate 2 is z. Referring to the figures shown in *Choosing the Element's Dimensionality*, the r-direction corresponds to the global X-direction in the $\theta = 0°$ plane and the negative global Z-direction in the $\theta = 90°$ plane, and the z-direction corresponds to the global Y-direction. Coordinate 1 must be greater than or equal to zero.

Degree of freedom 1 is u_r, degree of freedom 2 is u_z. The u_θ degree of freedom is an internal variable: you cannot control it.

References:

- *Choosing the Element's Dimensionality*
- *Solid (Continuum) Elements*
- **SOLID SECTION*

Element Types

Stress/Displacement Elements

CAXA4*N*	Bilinear, Fourier quadrilateral with 4 nodes per r–z plane
CAXA4H*N*	Bilinear, Fourier quadrilateral with 4 nodes per r–z plane, hybrid with constant Fourier pressure
CAXA4R*N*	Bilinear, Fourier quadrilateral with 4 nodes per r–z plane, reduced integration in r–z planes with hourglass control
CAXA4RH*N*	Bilinear, Fourier quadrilateral with 4 nodes per r–z plane, reduced integration in r–z planes, hybrid with constant Fourier pressure
CAXA8*N*	Biquadratic, Fourier quadrilateral with 8 nodes per r–z plane
CAXA8H*N*	Biquadratic, Fourier quadrilateral with 8 nodes per r–z plane, hybrid with linear Fourier pressure
CAXA8R*N*	Biquadratic, Fourier quadrilateral with 8 nodes per r–z plane, reduced integration in r–z planes
CAXA8RH*N*	Biquadratic, Fourier quadrilateral with 8 nodes per r–z plane, reduced integration in r–z planes, hybrid with linear Fourier pressure

Active Degrees of Freedom

1, 2

Additional Solution Variables

The bilinear elements have 4*N* and the biquadratic elements 8*N* additional variables relating to u_θ.

Element types CAXA4H*N* and CAXA4RH*N* have $1 + N$ additional variables relating to the pressure stress.

Element types CAXA8H*N* and CAXA8RH*N* have $3(1 + N)$ additional variables relating to the pressure stress.

Pore Pressure Elements

CAXA8P*N*	Biquadratic, Fourier quadrilateral with 8 nodes per *r*–*z* plane, bilinear Fourier pore pressure
CAXA8RP*N*	Biquadratic, Fourier quadrilateral with 8 nodes per *r*–*z* plane, bilinear Fourier pore pressure, reduced integration in *r*–*z* planes

Active Degrees of Freedom

1, 2, 8 at corner nodes

1, 2 at midside nodes

Additional Solution Variables

8*N* additional variables relating to u_θ.

Nodal Coordinates Required

r, *z*

Element Property Definition

Input File Usage: **SOLID SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Solid** as the section **Category** and **Homogeneous** as the section **Type**

Element-Based Loading

Even though the symmetry in the r–z plane at $\theta = 0, \pi$ allows the modeling of half of the initially axisymmetric structure, the loading must be specified as the total load on the full axisymmetric body. Consider, for example, a cylindrical shell loaded by a unit uniform axial force. To produce a unit load on a CAXA element with 4 modes, the nodal forces are 1/8, 1/4, 1/4, 1/4, and 1/8 at $\theta = 0$, $\pi/4$, $\pi/2$, $3\pi/4$, and π, respectively.

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Units: FL^{-3}
Description: Body force per unit volume in the global X-direction.

Load ID (*DLOAD): BZ
Units: FL^{-3}
Description: Body force per unit volume in the z-direction.

Load ID (*DLOAD): BXNU
Units: FL^{-3}
Description: Nonuniform body force in the global X-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BZNU
Units: FL^{-3}
Description: Nonuniform body force in the z-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): P*n*
Units: FL^{-2}
Description: Pressure on face n.

Load ID (*DLOAD): P*n*NU
Units: FL^{-2}
Description: Nonuniform pressure on face n with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): HP*n*
Units: FL^{-2}
Description: Hydrostatic pressure on face n, linear in the global Y-direction.

Foundations

Foundations are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): F*n*
Units: FL^{-3}
Description: Elastic foundation on face n.

Distributed Flows

Distributed flows are available for elements with pore pressure degrees of freedom. They are specified as described in *Coupled Pore Fluid Diffusion and Stress Analysis*.

Load ID (*FLOW/ *DFLOW): Q*n*
Units: $F^{-1}L^3T^{-1}$
Description: Seepage (outward normal flow) proportional to the difference between surface pore pressure and a reference sink pore pressure on face *n* (units of FL^{-2}).

Load ID (*FLOW/ *DFLOW): Q*n*D
Units: $F^{-1}L^3T^{-1}$
Description: Drainage-only seepage (outward normal flow) proportional to the surface pore pressure on face *n* only when that pressure is positive.

Load ID (*FLOW/ *DFLOW): Q*n*NU
Units: $F^{-1}L^3T^{-1}$
Description: Nonuniform seepage (outward normal flow) proportional to the difference between surface pore pressure and a reference sink pore pressure on face *n* (units of FL^{-2}) with magnitude supplied via user subroutine *FLOW*.

Load ID (*FLOW/ *DFLOW): S*n*
Units: LT^{-1}
Description: Prescribed pore fluid velocity (outward from the face) on face *n*.

Load ID (*FLOW/ *DFLOW): S*n*NU
Units: LT^{-1}
Description: Nonuniform prescribed pore fluid velocity (outward from the face) on face *n* with magnitude supplied via user subroutine *DFLOW*.

Element Output

The numerical integration with respect to θ employs the trapezoidal rule. There are $2(N+1)$ equally spaced integration planes in the element, including the $\theta = 0°$ and $\theta = 180°$ planes, with *N* being the number of Fourier modes. Consequently, the radial nodal forces corresponding to pressure loads applied in the circumferential direction are distributed in this direction in the ratio of $1:1$ in the 1 Fourier mode element, $1:2:1$ in the 2 Fourier mode element, and $1:2:2:2:1$ in the 4 Fourier mode element. The sum of these consistent nodal forces is equal to the integral of the applied pressure over 2π.

Output is as defined below unless a local coordinate system in the *r*–*z* plane is assigned to the element through the section definition (*Orientations*) in which case the components are in the local directions. These local directions rotate with the motion in large-displacement analysis. See *State storage* for details.

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11 Stress in the radial direction or in the local 1-direction.

S22 Stress in the axial direction or in the local 2-direction.

S33 Hoop direct stress.

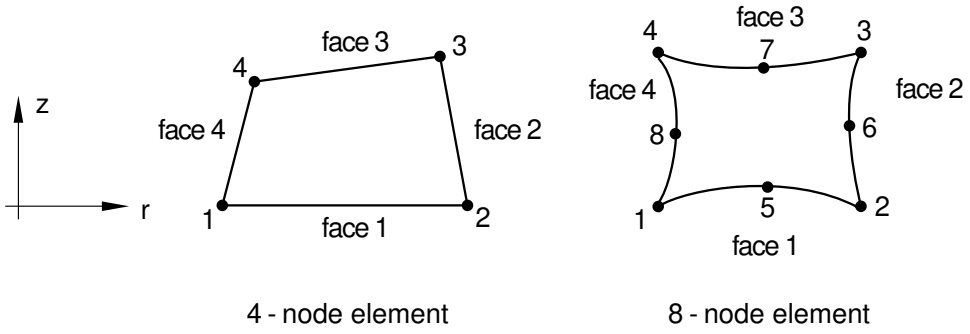
S12 Shear stress.

S13 Shear stress.

S23 Shear stress.

Node Ordering and Face Numbering on Elements

The node ordering in the first r–z plane of each element, at $\theta = 0$, is shown below. Each element must have N more planes of nodes defined, where N is the number of Fourier modes. The node ordering is the same in each plane. You can specify the nodes in each plane. Alternatively, you can specify the node ordering in the first r–z plane of an element, and Abaqus/Standard will generate all other nodes for the element by adding successively a constant offset to each node for each of the N planes of the element. By default, Abaqus/Standard uses an offset of 100000 (see *Element Definition*).



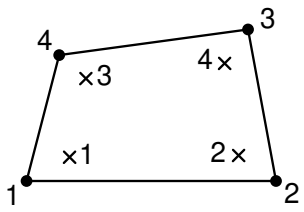
4 - node element 8 - node element

Table 1: Element faces

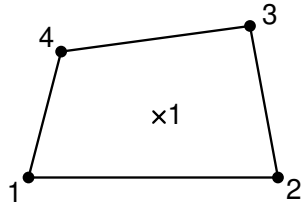
Face 1	1 – 2 face
Face 2	2 – 3 face
Face 3	3 – 4 face
Face 4	4 – 1 face

Numbering of Integration Points for Output

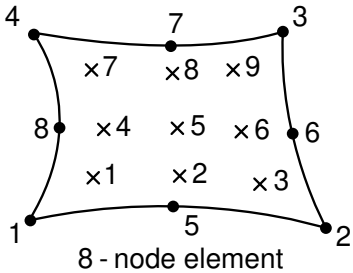
The integration points in the first r–z plane of integration, at $\theta = 0$, are shown below. The integration points follow in sequence at the r–z integration planes in ascending order of θ location.



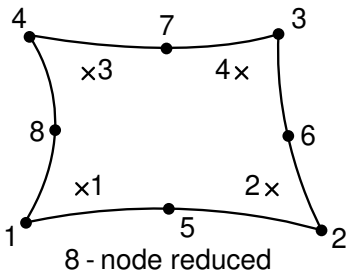
4 - node element



4 - node reduced integration element



8 - node element



8 - node reduced integration element

Infinite Elements

In this section:

- *Infinite Elements*
- *Infinite Element Library*

Infinite Elements

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Infinite Element Library*
- **SOLID SECTION*
- *Creating acoustic infinite sections*

Overview

Infinite elements:

- are used in boundary value problems defined in unbounded domains or problems in which the region of interest is small in size compared to the surrounding medium;
- are usually used in conjunction with finite elements;
- can have linear behavior only;
- provide stiffness in static solid continuum analyses; and
- provide “quiet” boundaries to the finite element model in dynamic analyses.

A solid section definition is used to define the section properties of infinite elements.

Typical Applications

The analyst is sometimes faced with boundary value problems defined in unbounded domains or problems in which the region of interest is small in size compared to the surrounding medium. Infinite elements are intended to be used for such cases in conjunction with first- and second-order planar, axisymmetric, and three-dimensional finite elements. Standard finite elements should be used to model the region of interest, with the infinite elements modeling the far-field region.

Choosing an Appropriate Element

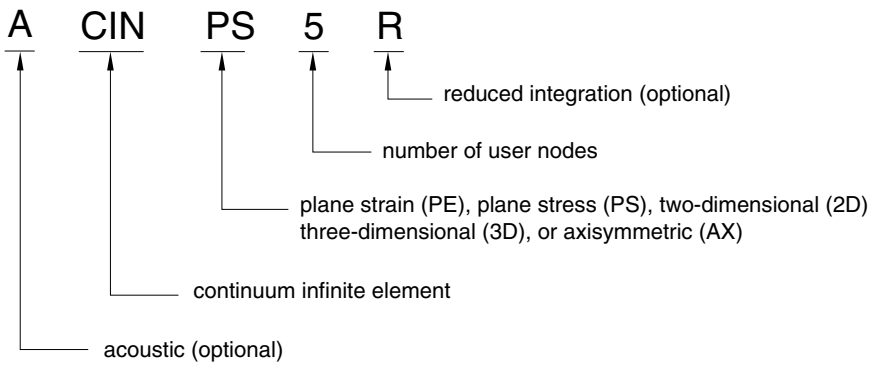
Plane stress, plane strain, three-dimensional, and axisymmetric infinite elements are available. Reduced-integration elements are also available in Abaqus/Standard.

Element type CIN3D18R is intended for use with the three-dimensional variable-number-of-node solids C3D15V, C3D27, and C3D27R in Abaqus/Standard.

Acoustic infinite elements are also available in Abaqus.

Naming Convention

Infinite elements in Abaqus are named as follows:



For example, CINAX4 is a 4-node, axisymmetric, infinite element.

Defining the Elements Section Properties

You use a solid section definition to define the section properties. You must associate these properties with a region of your model.

Input File Usage: **SOLID SECTION*, ELSET=*name*

where the ELSET parameter refers to a set of infinite elements.

Abaqus/CAE Usage: Only acoustic infinite sections are supported in Abaqus/CAE.

Property module:
Create Section: select **Other** as the section **Category** and **Acoustic infinite** as the section **Type**
Assign->Section: select regions

Defining the Thickness for Plane Strain and Plane Stress Elements

You define the thickness for plane strain and plane stress elements as part of the section definition. If you do not specify a thickness, unit thickness is assumed.

Input File Usage: **SOLID SECTION*
thickness

Abaqus/CAE Usage: Structural infinite sections are not supported in Abaqus/CAE.

Defining the Reference Point and Thickness for Acoustic Infinite Elements

For acoustic infinite elements you specify the thickness and the reference point. The thickness is ignored in three-dimensional and axisymmetric elements. You can prescribe the reference point either as a reference node on the section definition (see below) or directly by giving its coordinates on the data line following the thickness value. If both methods are used, the former takes precedence. If you do not define the reference point at all, an error message is issued.

The location of the reference point is used to determine the “radius” and “node ray” at each node of acoustic infinite elements, as shown in *Figure 1*.

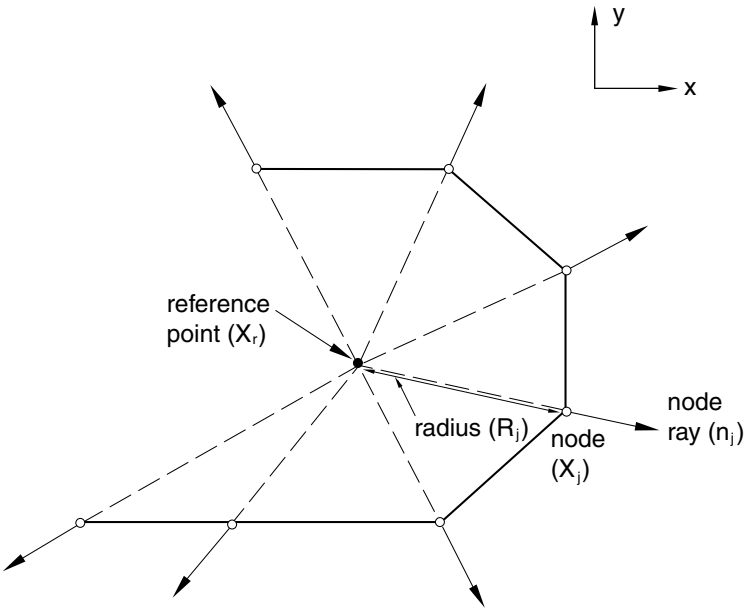


Figure 1: Reference point and node rays for acoustic infinite elements.

Each node ray is a unit vector in the direction of the line between the reference point and the node. These radii and rays are used in the formulation of acoustic infinite elements. The placement of the reference point is not extremely critical as long as it is near the center of the finite region enclosed by the infinite elements. If acoustic infinite elements are placed on the surface of a sphere, the optimal location for the reference point is the center of the sphere.

Acoustic infinite elements whose section properties are defined using a particular solid section definition should not have any nodes in common with acoustic infinite elements associated with a different solid section definition. This is to ensure a unique reference point (and, therefore, a unique “radius” and “node ray”) for each acoustic infinite element node.

The node rays are used to compute “cosine” values at each node of the infinite element interface. The “cosine” is equal to the smallest dot product of the unit node ray and the unit normals of all acoustic infinite element faces surrounding the node (see *Figure 2*). An error message is issued for negative values of “cosine.” Both the “radius” and “cosine” for all nodes of acoustic infinite elements are printed to the data (`.dat`) file as nodal (model) data. For details of how these quantities are used in the formulation, see *Acoustic infinite elements*.

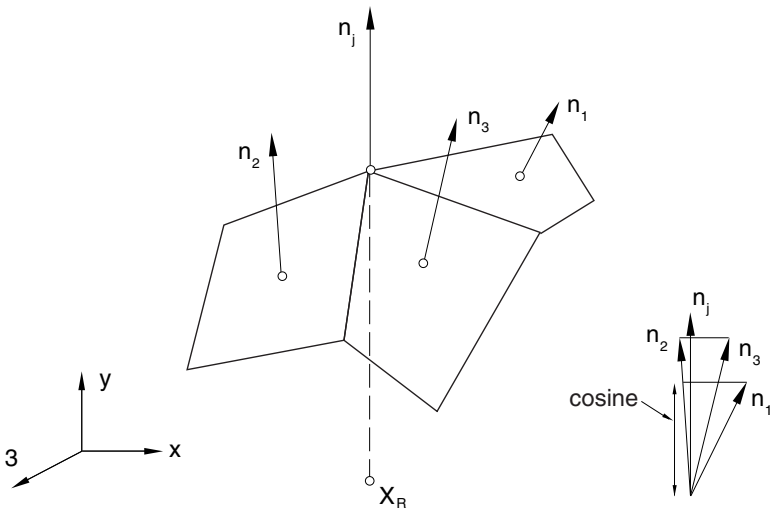


Figure 2: Defining the cosine for acoustic infinite elements.

Input File Usage: **SOLID SECTION*, REF NODE=*node number or node set name*
thickness

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Acoustic infinite** as the section **Type**: **Plane stress/strain thickness**: *thickness*

Acoustic infinite sections must be assigned to regions of parts that have a reference point associated with them. To define the reference point:

Part module or Property module: **Tools->Reference Point**: select reference point

Defining the Order of Interpolation for Acoustic Infinite Elements

For acoustic infinite elements the variation of the acoustic field in the infinite direction is given by functions that are members of a set of 10 ninth-order polynomials (for further details, see *Acoustic infinite elements*). The members of this set are constructed to correspond to the Legendre modes of a sphere; that is, if infinite elements are placed on a sphere and if tangential refinement is adequate, an *i*th order acoustic infinite element will absorb waves associated with the $(i-1)$th Legendre mode. The computational cost involved in using all 10 members in this set of polynomials to resolve the variation of the acoustic field in the infinite direction may be significant in certain applications in Abaqus/Explicit. In such cases you may wish to include only the first few members of the set, although you should be aware of the possibility of degraded accuracy (i.e., increased reflection at acoustic infinite elements) due to using a reduced set of polynomials. In Abaqus/Explicit you can specify the number, *N*, of ninth-order polynomials to be used. By default, all 10 members of the set will be used; all 10 are always used in Abaqus/Standard. Specifying a value less than 10 would result in the first *N* members of the set being used to model the variation of the acoustic field in the infinite direction.

Input File Usage: **SOLID SECTION*, ORDER=*N*

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Acoustic infinite** as the section **Type**: **Order**: *N*

Assigning a Material Definition to a Set of Infinite Elements

You must associate a material definition with each infinite element section definition. Optionally, you can associate a material orientation definition with the section (see *Orientations*).

The solution in the far field is assumed to be linear, so that only linear behavior can be associated with infinite elements (*Linear Elastic Behavior*). In dynamic analysis the material response in the infinite elements is also assumed to be isotropic.

In Abaqus/Explicit the material properties assigned to the infinite elements must match the material properties of the adjacent finite elements in the linear domain.

Only an acoustic medium material (*Acoustic Medium*) is valid for acoustic infinite elements.

Input File Usage: **SOLID SECTION*, MATERIAL=*name*, ORIENTATION=*name*

Abaqus/CAE Usage: Only acoustic infinite sections are supported in Abaqus/CAE.

Property module:
Create Section: select **Other** as the section **Category** and **Acoustic infinite** as the section **Type**: **Material:** *name*
Assign->Material Orientation: select regions
Assign->Section: select regions

Defining Nodes for Solid Medium Infinite Elements

The node numbering for infinite elements must be defined such that the first face is the face that is connected to the finite element part of the mesh.

The infinite element nodes that are not part of the first face are treated differently in explicit dynamic analysis than in other procedures. These nodes are located away from the finite element mesh in the infinite direction. The location of these nodes is not meaningful for explicit analysis, and loads and boundary conditions must not be specified using these nodes in explicit dynamic procedures. In other procedures these outer nodes are important in the element definition and can be used in load and boundary condition definitions.

Except for explicit procedures, the basis of the formulation of the solid medium elements is that the far-field solution along each element edge that stretches to infinity is centered about an origin, called the “pole.” For example, the solution for a point load applied to the boundary of a half-space has its pole at the point of application of the load. It is important to choose the position of the nodes in the infinite direction appropriately with respect to the pole. The second node along each edge pointing in the infinite direction must be positioned so that it is twice as far from the pole as the node on the same edge at the boundary between the finite and the infinite elements. Three examples of this are shown in *Figure 3*, *Figure 4*, and *Figure 5*. In addition to this length consideration, you must specify the second nodes in the infinite direction such that the element edges in the infinite direction do not cross over, which would give nonunique mappings (see *Figure 6*). Abaqus will stop with an error message if such problems occur. A convenient way of defining these second nodes in the infinite direction is to project the original nodes from a pole node; see *Projecting the Nodes in the Old Set from a Pole Node*. The positions of the pole and of the nodes on the boundary between the finite and the infinite elements are used.

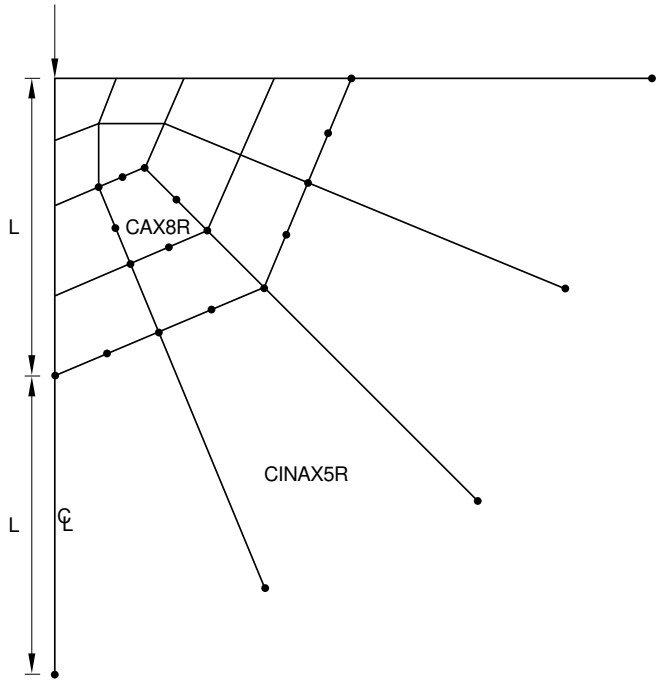


Figure 3: Point load on elastic half-space.

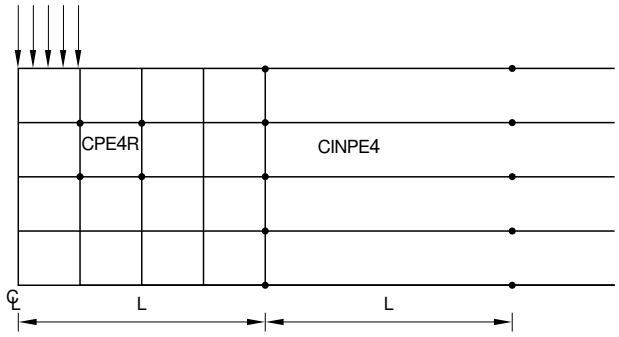


Figure 4: Strip footing on infinitely extending layer of soil.

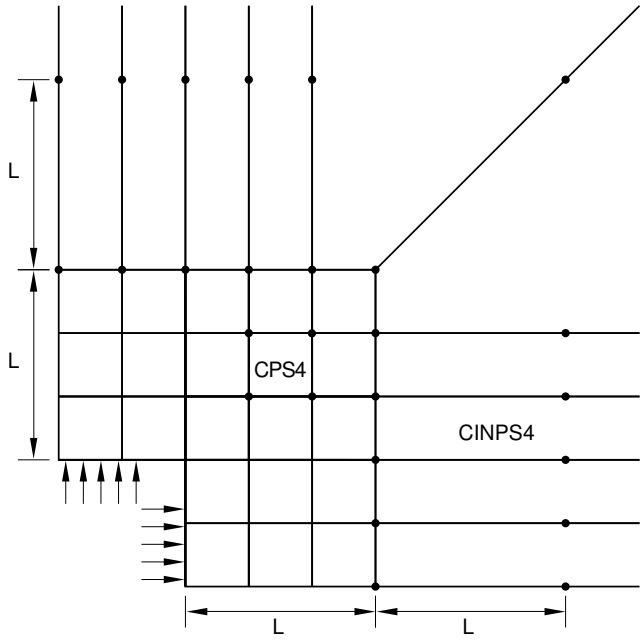


Figure 5: Quarter plate with square hole.

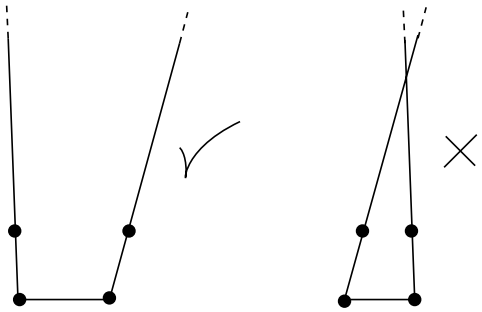


Figure 6: Examples of an acceptable and an unacceptable two-dimensional infinite element.

Defining Nodes for Acoustic Infinite Elements

The nodes of acoustic infinite elements need to be defined only for the face that is connected to the finite element part of the mesh. Additional nodes are generated internally by Abaqus in the direction of the "node ray" (see *Figure 1*). The node rays, which are discussed earlier in this section in the context of defining the reference point, define the sides of the acoustic infinite elements.

Using Solid Medium Infinite Elements in Plane Stress and Plane Strain Analyses

In plane stress and plane strain analyses when the loading is not self-equilibrating, the far-field displacements typically have the form $u = \ln(r)$, where r is distance from the origin. This form implies that the displacement approaches infinity as $r \to \infty$. Infinite elements will not provide a unique displacement solution for such cases. Experience shows, however, that they can still be used, provided that the displacement results are treated as having an arbitrary reference value. Thus, strain, stress, and *relative* displacements within the finite element part of the model will converge to unique values as the model is refined; the *total* displacements will depend on the size of the region modeled with finite elements. If the loading is self-equilibrating, the total displacements will also converge to a unique solution.

Using Solid Medium Infinite Elements in Dynamic Analyses

In direct-integration implicit dynamic response analysis (*Implicit Dynamic Analysis Using Direct Integration*), steady-state dynamic frequency domain analysis (*Direct-Solution Steady-State Dynamic Analysis*), matrix generation (*Generating Matrices as a Linear Analysis Step*), superelement generation (*Using Substructures*), and explicit dynamic analysis (*Explicit Dynamic Analysis*), infinite elements provide "quiet" boundaries to the finite element model through the effect of a damping matrix; the stiffness matrix of the element is suppressed. The elements do not provide any contribution to the eigenmodes of the system. The elements maintain the static force that was present at the start of the dynamic response analysis on this boundary; as a consequence, the far-field nodes in the infinite elements will not displace during the dynamic response.

During dynamic steps the infinite elements introduce additional normal and shear tractions on the finite element boundary that are proportional to the normal and shear components of the velocity of the boundary. These boundary damping constants are chosen to minimize the reflection of dilatational and shear wave energy back into the finite element mesh. This formulation does not provide perfect transmission of energy out of the mesh except in the case of plane body waves impinging orthogonally on the boundary in an isotropic medium. However, it usually provides acceptable modeling for most practical cases.

During dynamic response analysis the infinite elements hold the static stress on the boundary constant but do not provide any stiffness. Therefore, some rigid body motion of the region modeled will generally occur. This effect is usually small.

Optimizing the Transmission of Energy out of the Finite Element Mesh

For dynamic cases the ability of the infinite elements to transmit energy out of the finite element mesh, without trapping or reflecting it, is optimized by making the boundary between the finite and infinite elements as close as possible to being orthogonal to the direction from which the waves will impinge on this boundary. Close to a free surface, where Rayleigh waves may be important, or close to a material interface, where Love waves may be important, the infinite elements are most effective if they are orthogonal to the surface. (Rayleigh and Love waves are surface waves that decay with distance from the surface.)

For acoustic medium infinite elements, these general guidelines apply as well.

Defining an Initial Stress Field and Corresponding Body Force Field

In many applications, especially geotechnical problems, an initial stress field and a corresponding body force field must be defined. For standard elements you define the initial stress field as an initial condition (*Defining Initial Stresses*) and the corresponding body force field as a distributed load (*Distributed Loads*). The body force cannot be defined for infinite elements since the elements are of infinite extent. Therefore, Abaqus automatically inserts forces at the nodes of the infinite elements that cause those nodes to be in static equilibrium at the start of the analysis. These forces remain constant throughout the analysis. This capability allows the initial geostatic stress field to be defined in the infinite elements, but it does not check whether or not the geostatic stress field is reasonable. If the initial stress field is due to a body force loading (such as gravity loading), this loading must be held constant during the step. In multistep analyses it must be maintained constant over all steps.

You must remember that when infinite elements are used in conjunction with an initial stress condition, it is essential that the initial stress field be in equilibrium. In Abaqus/Standard any procedure that determines the initial static (steady-state) equilibrium conditions is suitable as the first step of the analysis; for example, static (*Static Stress Analysis*); geostatic stress field (*Geostatic Stress State*); coupled pore fluid diffusion/stress (*Coupled Pore Fluid Diffusion and Stress Analysis*); and steady-state fully coupled thermal-stress (*Fully Coupled Thermal-Stress Analysis*) steps can be used. To check for equilibrium in Abaqus/Explicit, perform an initial step with no loading (except for the body forces that created the initial stress field) and verify that the accelerations are small.

Infinite Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the infinite elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *Infinite Elements*
- **SOLID SECTION*

Element Types

Plane Strain Solid Continuum Infinite Elements

CINPE4	4-node linear, one-way infinite
CINPE5R$^{(S)}$	5-node quadratic, one-way infinite

Active Degrees of Freedom

1, 2

Additional Solution Variables

None.

Plane Stress Solid Continuum Infinite Elements

CINPS4	4-node linear, one-way infinite
CINPS5R$^{(S)}$	5-node quadratic, one-way infinite

Active Degrees of Freedom

1, 2

Additional Solution Variables

None.

3D Solid Continuum Infinite Elements

CIN3D8	8-node linear, one-way infinite
CIN3D12R$^{(S)}$	12-node quadratic, one-way infinite
CIN3D18R$^{(S)}$	18-node quadratic, one-way infinite

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

None.

Axisymmetric Solid Continuum Infinite Elements

CINAX4	4-node linear, one-way infinite
CINAX5R$^{(S)}$	5-node quadratic, one-way infinite

Active Degrees of Freedom

1, 2

Additional Solution Variables

None.

2D Acoustic Infinite Elements

ACIN2D2	2-node linear, acoustic infinite
ACIN2D3$^{(S)}$	3-node quadratic, acoustic infinite

Active Degrees of Freedom

8

3D Acoustic Infinite Elements

ACIN3D3	3-node linear, acoustic infinite triangular element
ACIN3D4	4-node linear, acoustic infinite quadrilateral element
ACIN3D6$^{(S)}$	6-node quadratic, acoustic infinite triangular element
ACIN3D8$^{(S)}$	8-node quadratic, acoustic infinite quadrilateral element

Active Degrees of Freedom

8

Axisymmetric Acoustic Infinite Elements

ACINAX2	2-node linear, acoustic infinite
ACINAX3$^{(S)}$	3-node quadratic, acoustic infinite

Active Degrees of Freedom

8

Nodal Coordinates Required

Plane stress and plane strain solid continuum elements: *X*, *Y*

2D acoustic elements: *X*, *Y*

3D solid continuum and acoustic elements: *X*, *Y*, *Z*

Axisymmetric solid continuum and acoustic elements: *r*, *z*

Normal directions are not specified at nodes used in acoustic infinite elements; they will be computed automatically. See *Infinite Elements* for details.

Element Property Definition

For two-dimensional, plane strain, and plane stress elements, you must provide the thickness of the elements; by default, unit thickness is assumed.

For three-dimensional and axisymmetric solid elements, you do not need to specify a thickness.

For acoustic elements, you must specify the reference point in addition to the thickness.

Input File Usage:

**SOLID SECTION*

Abaqus/CAE Usage: Only acoustic infinite sections are supported in Abaqus/CAE.

Property module: **Create Section**: select **Other** as the section **Category** and **Acoustic infinite** as the section **Type**

Element-Based Loading

None.

Element Output

Stress, Strain, and Other Tensor Components

No output is available from Abaqus/Explicit for infinite elements. Stress and other tensors (including strain tensors) are available from Abaqus/Standard for infinite elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11
XX direct stress or radial stress for axisymmetric elements.

S22
YY direct stress or axial stress for axisymmetric elements.

S33
ZZ direct stress (not available for plane stress elements) or hoop stress for axisymmetric elements.

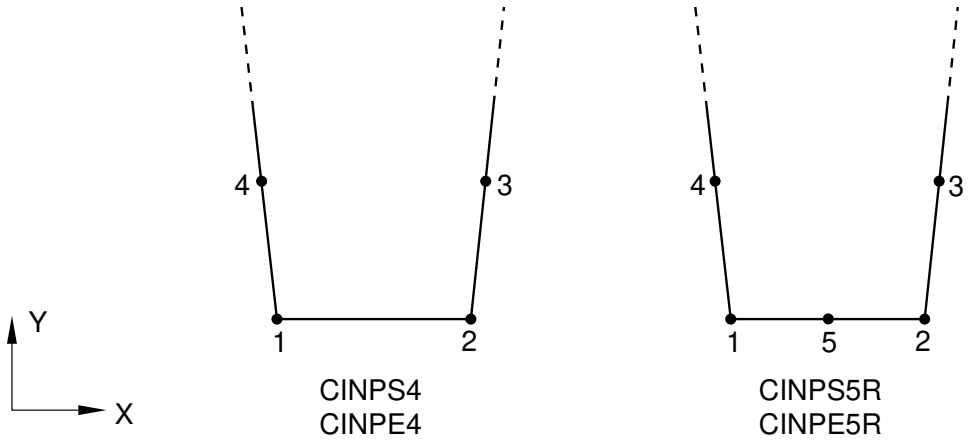
S12
XY shear stress or shear stress for axisymmetric elements.

S13
XZ shear stress (not available for plane stress, plane strain, and axisymmetric elements).

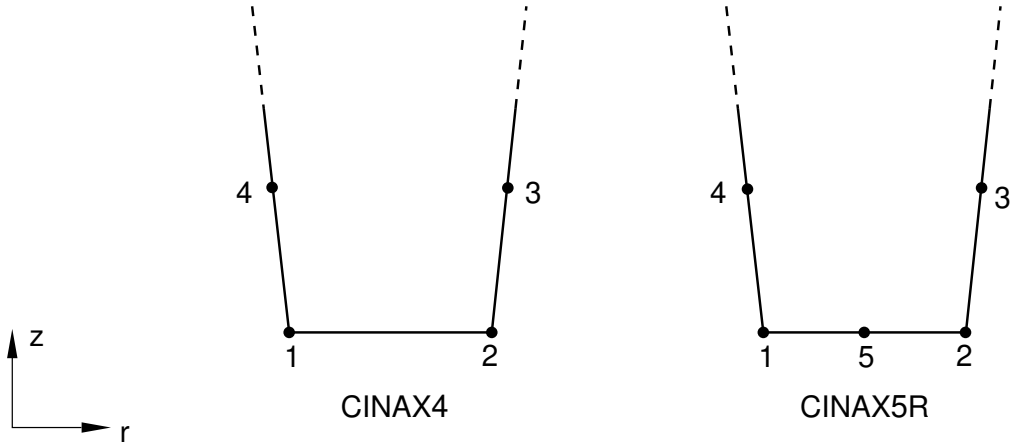
S23
YZ shear stress (not available for plane stress, plane strain, and axisymmetric elements).

Node Ordering and Face Numbering on Elements

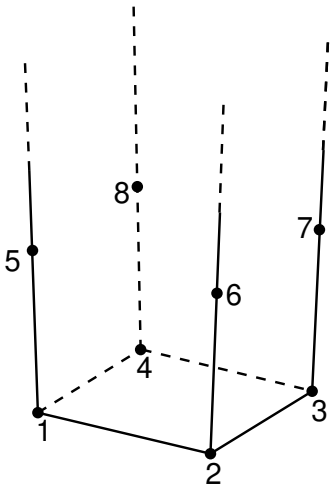
Plane Stress and Plane Strain Solid Continuum Elements



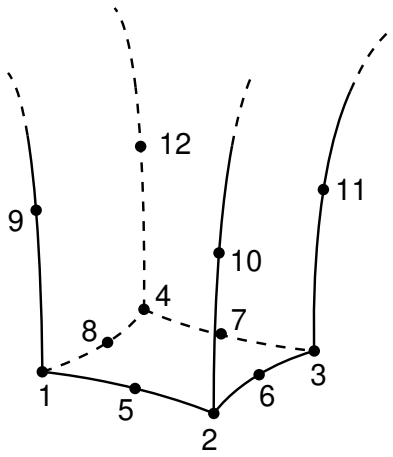
Axisymmetric Solid Continuum Elements



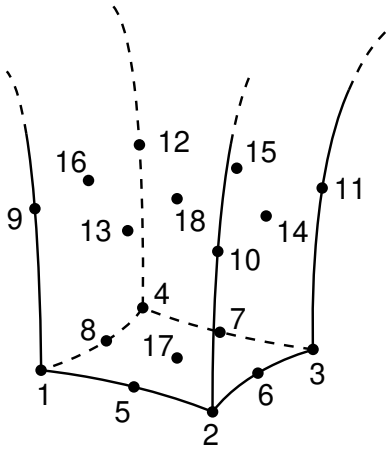
Three-Dimensional Solid Continuum Elements

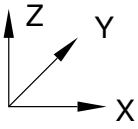


CIN3D8



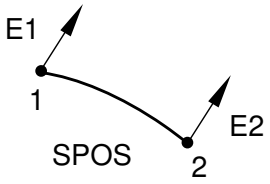
CIN3D12R



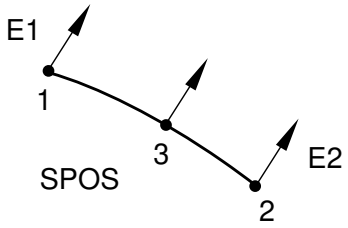


CIN3D18R

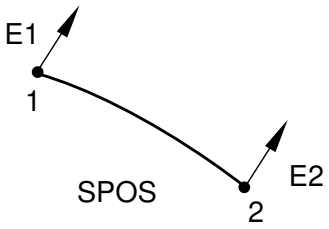
Two-Dimensional and Axisymmetric Acoustic Infinite Elements



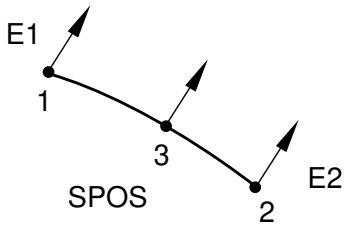
ACIN2D2



ACIN2D3

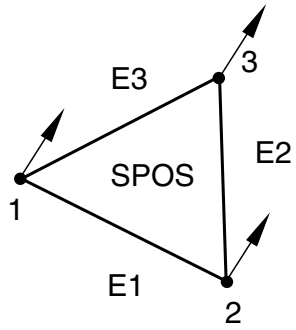


ACINAX2

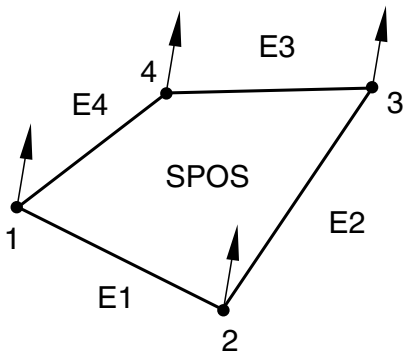


ACINAX3

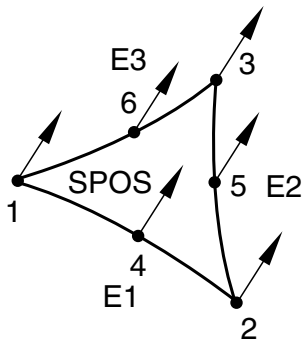
Three-Dimensional Acoustic Infinite Elements



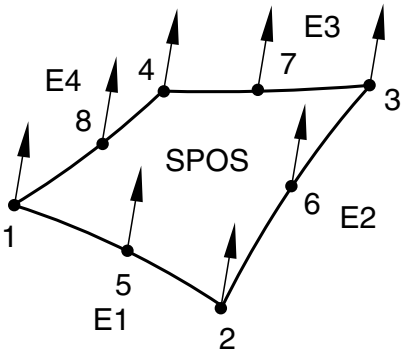
ACIN3D3



ACIN3D4



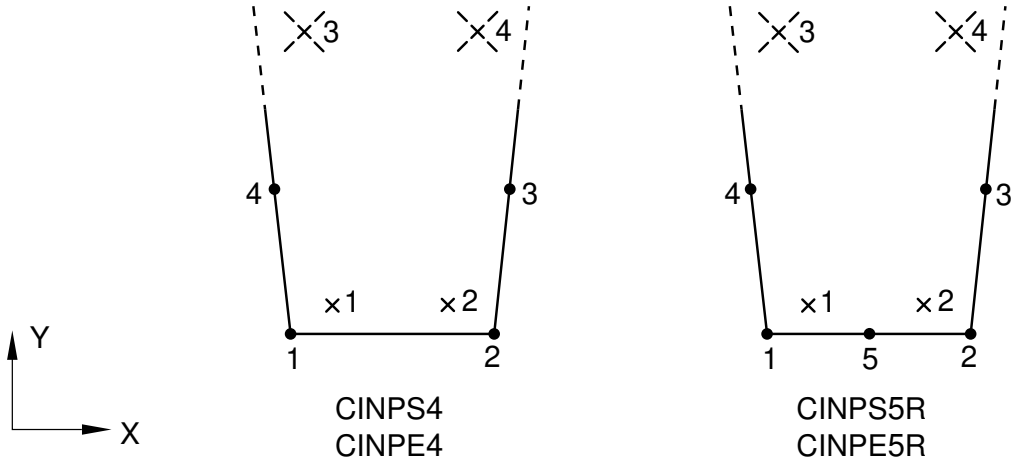
ACIN3D6



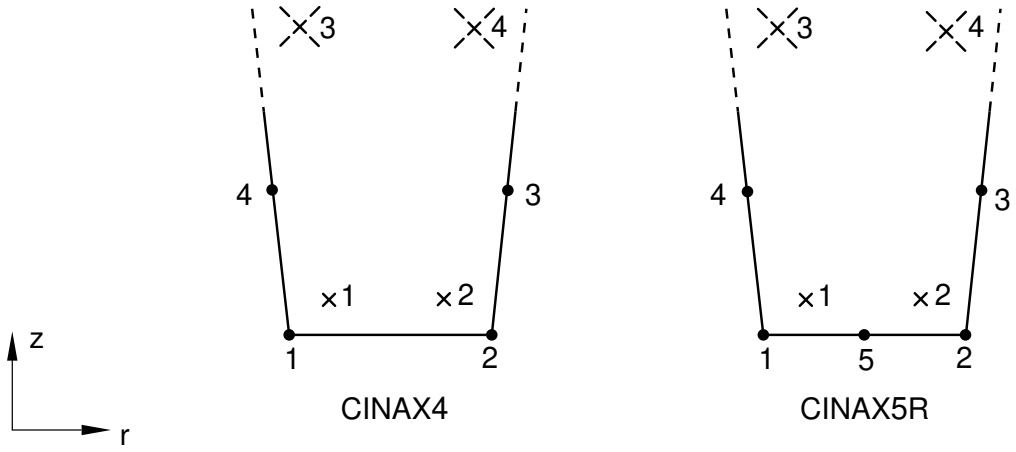
ACIN3D8

Numbering of Integration Points for Output

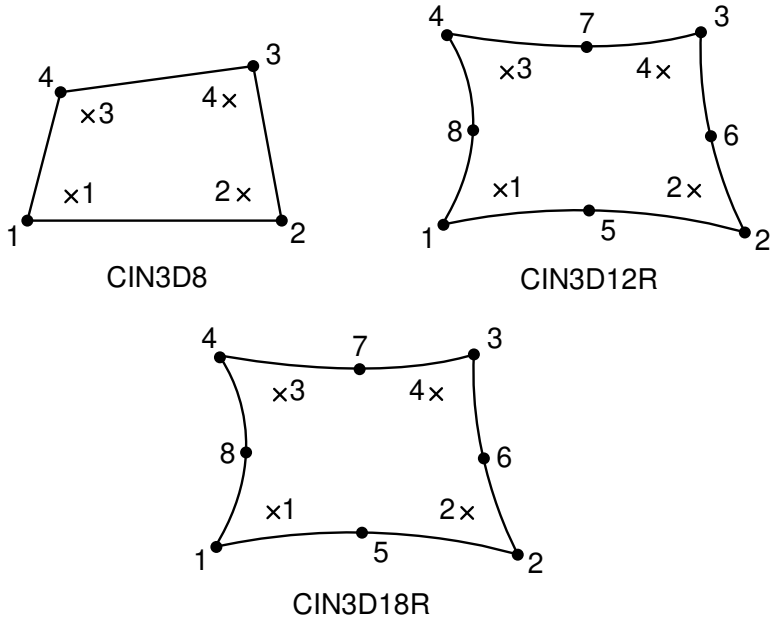
Plane Stress and Plane Strain Solid Continuum Elements



Axisymmetric Solid Continuum Elements



Three-Dimensional Solid Continuum Elements



This shows the scheme in the layer closest to the 1–2–3–4 face. The integration points in the second layer are numbered consecutively.

Warping Elements

In this section:

- *Warping Elements*
- *Warping Element Library*

Warping Elements

Products: Abaqus/Standard

References:

- *Meshed Beam Cross-Sections*
- **SOLID SECTION*

Overview

Warping elements:

- are used to model an arbitrarily shaped beam cross-section profile for use with Timoshenko beams;
- are used in conjunction with the beam section generation procedure described in *Meshed Beam Cross-Sections*; and
- model linear elastic behavior only.

Typical Applications

Warping elements are special-purpose elements that are used to discretize a two-dimensional model of a beam cross-section. This two-dimensional cross-section model is used in Abaqus/Standard to calculate the out-of-plane and, optionally, the in-plane warping displacements, as well as relevant sectional stiffness and mass properties that are required in a subsequent beam analysis in either Abaqus/Standard or Abaqus/Explicit. Applications include any structure whose overall behavior is beam-like, yet the cross-section is non-standard or includes multiple materials. Examples include the cross-section of a ship for performing whipping analysis, a beam model of an airfoil-shaped rotor blade or wing, a laminated I-beam, etc.

Choosing an Appropriate Element

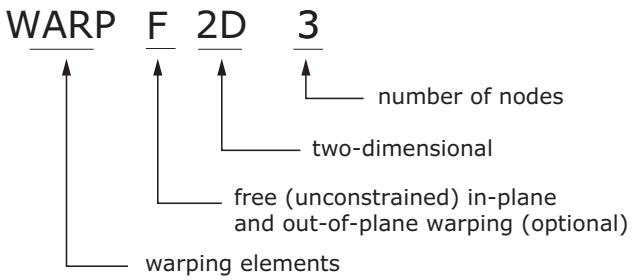
To mesh an arbitrarily shaped solid beam cross-section Abaqus/Standard offers two element classes:

- 1-DOF warping elements have one degree of freedom per node representing the out-of-plane warping displacement.
- 3-DOF warping elements have two additional degrees of freedom per node representing the in-plane warping displacements.

Adjacent elements in the cross-sectional mesh must share common nodes; mesh refinement using multi-point constraints is not allowed. The mesh can contain either 1-DOF warping elements or 3-DOF warping elements; you cannot mix the two element classes in the same analysis.

Naming Convention

Warping elements are named as follows:



For example, WARP2D4 is 4-node warping element in two dimensions.

Defining the Elements Section Properties

You use a solid section definition to define the section properties. You must associate these properties with a region of your model. No additional data are necessary.

Input File Usage: **SOLID SECTION*, ELSET=*name*

where the ELSET parameter refers to a set of warping elements.

Assigning a Material Definition to a Set of Warping Elements

You must associate a linear elastic material definition with each warping element section definition. Optionally, you can associate a material orientation definition with the section (see *Orientations*).

Only isotropic linear elasticity (*Defining Isotropic Elasticity*) or orthotropic linear elasticity for warping elements (*Defining Orthotropic Elasticity for 1-DOF Warping Elements*) are valid material models for 1-DOF warping elements.

For 3-DOF warping elements, you can define the following types of linear elasticity:

- Isotropic behavior (*Defining Isotropic Elasticity*)
- Orthotropic behavior (*Defining Orthotropic Elasticity by Specifying the Engineering Constants*, *Defining Transversely Isotropic Elasticity*, and *Defining Orthotropic Elasticity by Specifying the Terms in the Elastic Stiffness Matrix*)
- Fully anisotropic behavior (*Defining Fully Anisotropic Elasticity*)

Input File Usage: **SOLID SECTION*, ELSET=*name*, MATERIAL=*name*, ORIENTATION=*name*

Warping Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the warping elements available in Abaqus/Standard.

References:

- *Meshed Beam Cross-Sections*
- **SOLID SECTION*

Element Types

1-DOF Warping Elements

WARP2D3	3-node linear two-dimensional warping element
WARP2D4	4-node bilinear two-dimensional warping element

Active Degrees of Freedom

3, representing the out-of-plane warping displacement

Additional Solution Variables

None.

3-DOF Warping Elements

WARPF2D3	3-node linear two-dimensional warping element with free (unconstrained) in-plane and out-of-plane warping
WARPF2D4	4-node bilinear two-dimensional warping element with free (unconstrained) in-plane and out-of-plane warping
WARPF2D6	6-node quadratic two-dimensional warping element with free (unconstrained) in-plane and out-of-plane warping
WARPF2D8	8-node quadratic two-dimensional warping element with free (unconstrained) in-plane and out-of-plane warping

Active Degrees of Freedom

1, 2, representing the in-plane warping displacements

3, representing the out-of-plane warping displacement

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*

Element Property Definition

Input File Usage: **SOLID SECTION*

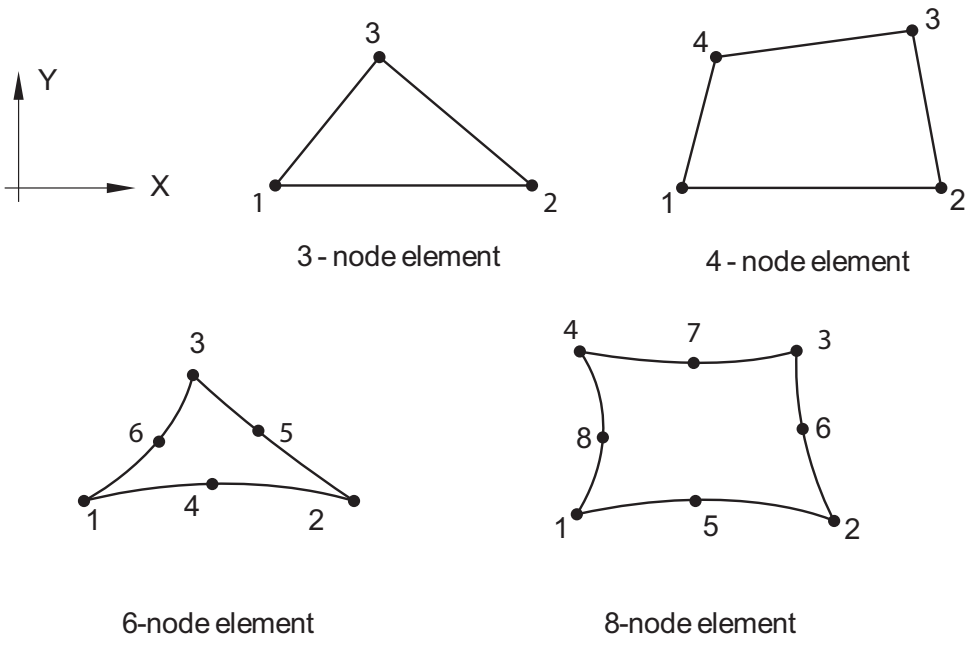
Element-Based Loading

There is no loading for these element types.

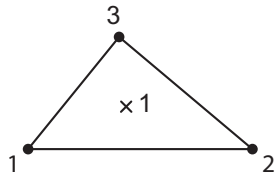
Element Output

The two-dimensional warping elements are used to calculate the out-of-plane warping and, for 3-DOF warping elements, the in-plane warping for beams using a meshed cross-section. One warping function is computed for 1-DOF warping elements, and six warping functions are computed for 3-DOF warping elements. You can view warping functions in the Visualization module of Abaqus/CAE. At the element integration points, the derivatives of the warping functions are used to calculate the shear strain and stress due to torsion (for 1-DOF warping elements) or the full three-dimensional strain and stress due to extension, shear, bending, and torsion (for 3-DOF warping elements).

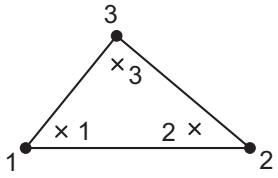
Node Ordering on Elements



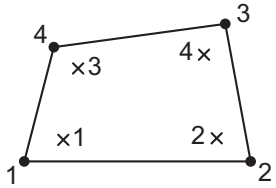
Numbering of Integration Points for Output



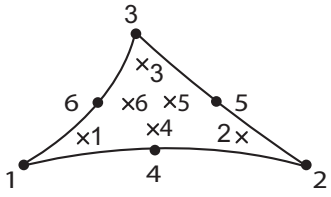
3 - node element
WARP2D3



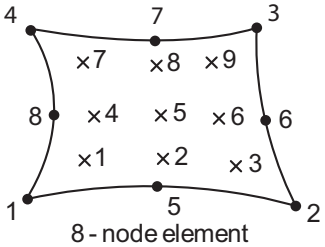
3-node element
WARPF2D3



4 - node element
WARP2D4
WARPF2D4



6 - node element
WARPF2D6



8 - node element
WARPF2D8

Structural Elements

In this section:

- *Membrane Elements*
- *Truss Elements*
- *Beam Elements*
- *Frame Elements*
- *Elbow Elements*
- *Shell Elements*
- *Shear Panel Elements*

Membrane Elements

In this section:

- *Membrane Elements*
- *General Membrane Element Library*
- *Cylindrical Membrane Element Library*
- *Axisymmetric Membrane Element Library*

Membrane Elements

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *General Membrane Element Library*
- *Cylindrical Membrane Element Library*
- *Axisymmetric Membrane Element Library*
- **MEMBRANE SECTION*
- **NODAL THICKNESS*
- **DISTRIBUTION*
- **HOURGLASS STIFFNESS*
- *Creating membrane sections*

Overview

Membrane elements:

- are surface elements that transmit in-plane forces only (no moments); and
- have no bending stiffness.

Typical Applications

Membrane elements are used to represent thin surfaces in space that offer strength in the plane of the element but have no bending stiffness; for example, the thin rubber sheet that forms a balloon. In addition, they are often used to represent thin stiffening components in solid structures, such as a reinforcing layer in a continuum. (If the reinforcing layer is made up of chords, rebar should be used. See *Defining Rebar as an Element Property*.)

Choosing an Appropriate Element

In addition to the general membrane elements available in both Abaqus/Standard and Abaqus/Explicit, cylindrical membrane elements and axisymmetric membrane elements are available in Abaqus/Standard only.

General Membrane Elements

General membrane elements should be used in three-dimensional models in which the deformation of the structure can evolve in three dimensions.

Cylindrical Membrane Elements

Cylindrical membrane elements are available in Abaqus/Standard for precise modeling of regions in a structure with circular geometry, such as a tire. The elements make use of trigonometric functions to interpolate displacements along the circumferential direction and use regular isoparametric interpolation in the radial or cross-sectional plane. They use three nodes along the circumferential direction and can span a 0 to 180° segment. Elements with both first-order and second-order interpolation in the cross-sectional plane are available.

The geometry of the element is defined by specifying nodal coordinates in a global Cartesian system. The default nodal output is also provided in a global Cartesian system. Output of stress, strain, and other material point quantities is done in a corotational system that rotates with the average material rotation.

The cylindrical elements can be used in the same mesh with regular elements. In particular, regular membrane elements can be connected directly to the nodes on the cross-sectional edge of cylindrical elements. For example, any edge of an M3D4 element can share nodes with the cross-sectional edges of an MCL6 element.

Compatible cylindrical solid elements (*Cylindrical Solid Element Library*) and surface elements with rebar (*Surface Elements*) are available for use with cylindrical membrane elements.

Axisymmetric Membrane Elements

The axisymmetric membrane elements available in Abaqus/Standard are divided into two categories: those that do not allow twist about the symmetry axis and those that do. These elements are referred to as the regular and generalized axisymmetric membrane elements, respectively.

The generalized axisymmetric membrane elements (axisymmetric membrane elements with twist) allow a circumferential component of loading or material anisotropy, which may cause twist about the symmetry axis. Both the circumferential load component and material anisotropy are independent of the circumferential coordinate θ. Since there is no dependence of the loading or the material on the circumferential coordinate, the deformation is axisymmetric.

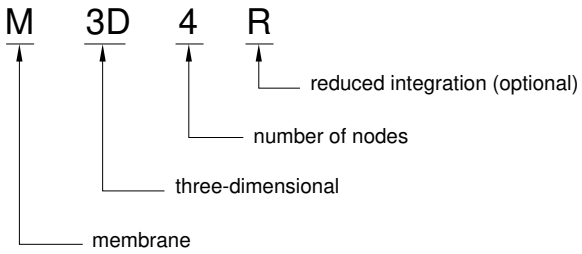
The generalized axisymmetric membrane elements cannot be used in dynamic or eigenfrequency extraction procedures.

Naming Convention

The naming convention for membrane elements depends on the element dimensionality.

General Membrane Elements

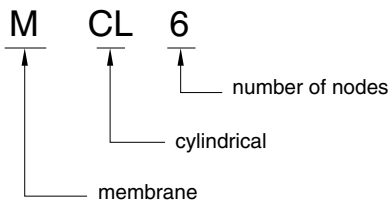
General membrane elements in Abaqus are named as follows:



For example, M3D4R is a three-dimensional, 4-node membrane element with reduced integration.

Cylindrical Membrane Elements

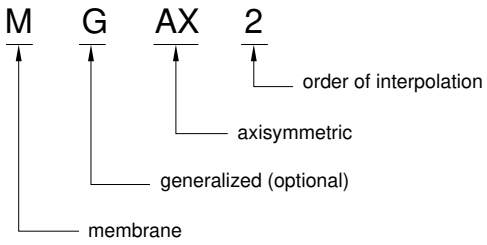
Cylindrical membrane elements in Abaqus/Standard are named as follows:



For example, MCL6 is a 6-node cylindrical membrane element with circumferential interpolation.

Axisymmetric Membrane Elements

Axisymmetric membrane elements in Abaqus/Standard are named as follows:



For example, MAX2 is a regular axisymmetric, quadratic-interpolation membrane element.

Element Normal Definition

The “top” surface of a membrane is the surface in the positive normal direction (defined below) and is called the SPOS face for contact definition. The “bottom” surface is in the negative direction along the normal and is called the SNEG face for contact definition.

General Membrane Elements

For general membrane elements the positive normal direction is defined by the right-hand rule going around the nodes of the element in the order that they are specified in the element definition. See *Figure 1*.

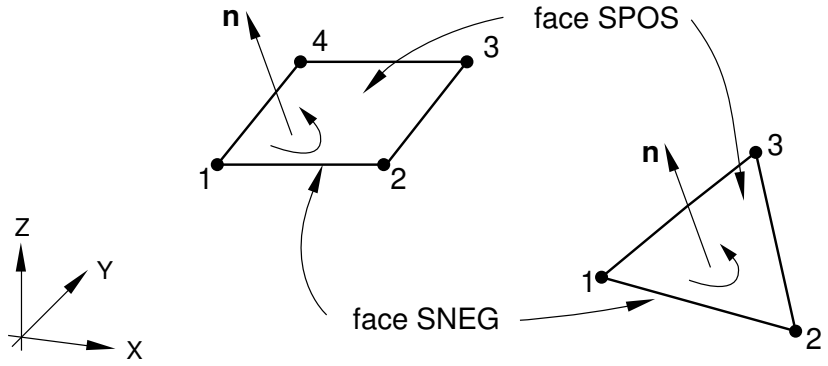


Figure 1: Positive normals for general membranes.

Cylindrical Membrane Elements

For cylindrical membrane elements the positive normal direction is defined by the right-hand rule going around the nodes of the element in the order that they are specified in the element definition. See *Figure 2*.

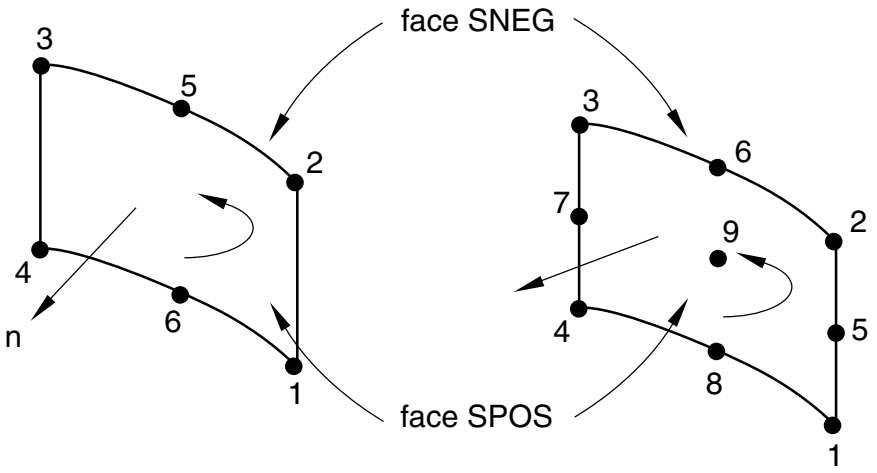


Figure 2: Positive normals for cylindrical membranes.

Axisymmetric Membrane Elements

For axisymmetric membrane elements the positive normal is defined by a 90° counterclockwise rotation from the direction going from node 1 to node 2. See *Figure 3*.

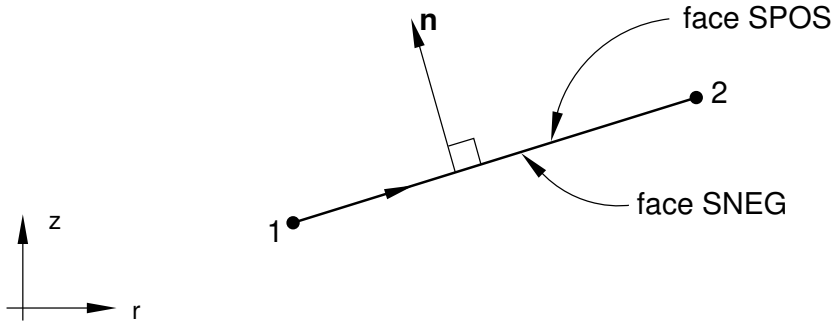


Figure 3: Positive normals for axisymmetric membranes.

Defining the Elements Section Properties

You use a membrane section definition to define the section properties. You must associate these properties with a region of your model.

Input File Usage: **MEMBRANE SECTION*, ELSET=*name*

where the ELSET parameter refers to a set of membrane elements.

Abaqus/CAE Usage: Property module:
Create Section: select **Shell** as the section **Category** and **Membrane** as the section **Type**
Assign->Section: select regions

Defining a Constant Section Thickness

You can define a constant section thickness as part of the section definition.

Input File Usage: **MEMBRANE SECTION*, ELSET=*name*
thickness

Abaqus/CAE Usage: Property module: **Create Section**: select **Shell** as the section **Category** and **Membrane** as the section **Type**: **Membrane thickness:** *thickness*

Defining a Variable Thickness Using Distributions

In Abaqus/Standard you can define a spatially varying thickness for membranes using a distribution (*Distribution Definition*).

The distribution used to define membrane thickness must have a default value. The default thickness is used by any membrane element assigned to the membrane section that is not specifically assigned a value in the distribution.

If the membrane thickness is defined for a membrane section with a distribution, nodal thicknesses cannot be used for that section definition.

Input File Usage: Use the following option to define a spatially varying thickness:

**MEMBRANE SECTION*, MEMBRANE THICKNESS=*distribution name*

Defining a Continuously Varying Thickness

Alternatively, you can define a continuously varying thickness over the element. In this case any constant section thickness you specify will be ignored, and the section thickness will be interpolated from the specified nodal values (see *Nodal Thicknesses*). The thickness must be defined at all nodes connected to the element.

If the membrane thickness is defined for a membrane section with a distribution, nodal thicknesses cannot be used for that section definition.

Input File Usage: Use both of the following options:

**MEMBRANE SECTION*, NODAL THICKNESS
**NODAL THICKNESS*

Abaqus/CAE Usage: Continuously varying membrane thicknesses are not supported in Abaqus/CAE.

Assigning a Material Definition to a Set of Membrane Elements

You must associate a material definition with each membrane section definition. Optionally, you can associate a material orientation definition with the section (see *Orientations*). An arbitrary material orientation is valid only for general membrane elements and axisymmetric membrane elements with twist. You can define other directions by defining a local orientation, except for MAX1 and MAX2 elements (*Axisymmetric Membrane Element Library*), which do not support orientations.

In Abaqus/Standard if the orientation assigned to a membrane section is defined with distributions, spatially varying local coordinate systems are applied to all membrane elements associated with the membrane section. A default local coordinate system (as defined by the distributions) is applied to any membrane element that is not specifically included in the associated distribution.

Input File Usage: **MEMBRANE SECTION*, MATERIAL=*name*, ORIENTATION=*name*

Abaqus/CAE Usage: Property module:
Create Section: select **Shell** as the section **Category** and **Membrane** as the section **Type**: **Material:** *name*
Assign->Material Orientation

Specifying How the Membrane Thickness Changes with Deformation

You can define how the membrane thickness will change with deformation by specifying a nonzero value for the section Poisson's ratio that will allow for a change in the thickness of the membrane as a function of the in-plane strains in geometrically nonlinear analysis (see *Defining an Analysis*).

Alternatively in Abaqus/Explicit, you can choose to have the thickness change computed through integration of the thickness-direction strain that is based on the element material definition and the plane stress condition.

The value of the effective Poisson's ratio for the section must be between −1.0 and 0.5. By default, the section Poisson's ratio is 0.5 in Abaqus/Standard to enforce incompressibility of the element; in Abaqus/Explicit the default thickness change is based on the element material definition.

A section Poisson's ratio of 0.0 means that the thickness will not change. Values between 0.0 and 0.5 mean that the thickness changes proportionally between the limits of no thickness change and incompressibility, respectively. A negative value of the section Poisson's ratio will result in an increase of the section thickness in response to tensile strains.

Input File Usage: Use one of the following options:

**MEMBRANE SECTION*, POISSON=ν_{eff}
**MEMBRANE SECTION*, POISSON=MATERIAL (available in Abaqus/Explicit only)

Abaqus/CAE Usage: Property module: **Create Section**: select **Shell** as the section **Category** and **Membrane** as the section **Type**: **Section Poisson's ratio: Use analysis default** or **Specify value:** ν_{eff}

Specifying Nondefault Hourglass Control Parameters for Reduced-Integration Membrane Elements

See *Methods for Suppressing Hourglass Modes* for more information about hourglass control.

Specifying a Nondefault Hourglass Control Formulation or Scale Factors

You can specify a nondefault hourglass control formulation or scale factors for reduced-integration membrane elements. The nondefault enhanced hourglass control formulation is available only for M3D4R elements.

Input File Usage: Use the following option to specify a nondefault hourglass control formulation in a section control definition:

**SECTION CONTROLS*, NAME=*name*,
HOURGLASS=*hourglass_control_formulation*

Use the following option to associate the section control definition with the membrane section:

**MEMBRANE SECTION*, CONTROLS=*name*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Hourglass control:** *hourglass_control_formulation*

Specifying Nondefault Hourglass Stiffness Factors

In Abaqus/Standard you can specify nondefault hourglass stiffness factors based on the default total stiffness approach for reduced-integration general membrane elements. These stiffness factors are ignored for axisymmetric membrane elements. There are no hourglass stiffness factors or scale factors for the nondefault enhanced hourglass control formulation.

Input File Usage: Use both of the following options:

**MEMBRANE SECTION*
**HOURGLASS STIFFNESS*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Hourglass stiffness: Specify**

Using Membrane Elements in Large-Displacement Implicit Analyses

Buckling can occur in Abaqus/Standard if a membrane structure is subject to compressive loading in a large-displacement analysis, causing out-of-plane deformation. Since a stress-free flat membrane has no stiffness perpendicular to its plane, out-of-plane loading will cause numerical singularities and convergence difficulties. Once some out-of-plane deformation has developed, the membrane will be able to resist out-of-plane loading.

In some cases loading the membrane elements in tension or adding initial tensile stress can overcome the numerical singularities and convergence difficulties associated with out-of-plane loading. However, you must choose the magnitude of the loading or initial stress such that the final solution is unaffected.

Using Membrane Elements in Abaqus/Standard Contact Analyses

Element types M3D8 and M3D8R are converted automatically to element types M3D9 and M3D9R, respectively, if a secondary surface on a contact pair is attached to the element.

General Membrane Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the general membrane elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *Membrane Elements*
- **NODAL THICKNESS*
- **MEMBRANE SECTION*

Element Types

M3D3	3-node triangle
M3D4	4-node quadrilateral
M3D4R	4-node quadrilateral, reduced integration, hourglass control
M3D6$^{(S)}$	6-node triangle
M3D8$^{(S)}$	8-node quadrilateral
M3D8R$^{(S)}$	8-node quadrilateral, reduced integration
M3D9$^{(S)}$	9-node quadrilateral
M3D9R$^{(S)}$	9-node quadrilateral, reduced integration, hourglass control

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage:

**MEMBRANE SECTION*

In addition, use the following option for variable thickness membranes:

**NODAL THICKNESS*

Abaqus/CAE Usage: Property module: **Create Section**: select **Shell** as the section **Category** and **Membrane** as the section **Type**

You cannot define variable thickness membranes in Abaqus/CAE.

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in the global *X*-direction.

Load ID (*DLOAD): BY
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in the global *Y*-direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in the global *Z*-direction.

Load ID (*DLOAD): BXNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in the global *X*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BYNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in the global *Y*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}

Description: Nonuniform body force in the global *Z*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): CENT (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-4} ($ML^{-3}T^{-2}$)

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): CENTRIF (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): CORIO (S)
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-4}T$ ($ML^{-3}T^{-1}$)

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit volume, ω is the angular velocity). The load stiffness due to Coriolis loading is not accounted for in direct steady-state dynamic analysis.

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): HP (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Hydrostatic pressure applied to the element reference surface and linear in global *Z*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure applied to the element reference surface. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): PNU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure applied to the element reference surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): ROTA (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}
Description: Rotary acceleration load (magnitude is input as α, where α is the rotary acceleration).

Load ID (*DLOAD): ROTDYNF (S)
Abaqus/CAE Load/Interaction: Not supported

Units: T^{-1}

Description: Rotordynamic load (magnitude is input as ω, where ω is the angular velocity).

Load ID (*DLOAD): SBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-5}T^2$
Description: Stagnation body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): SP (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T^2$
Description: Stagnation pressure applied to the element reference surface.

Load ID (*DLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element reference surface.

Load ID (*DLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): VBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T$
Description: Viscous body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): VP (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-3}T$
Description: Viscous surface pressure applied to the element reference surface. The pressure is proportional to the velocity normal to the element face and opposing the motion.

Foundations

Foundations are available only in Abaqus/Standard and are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): F (S)
Abaqus/CAE Load/Interaction: Elastic foundation
Units: FL^{-3}

Description: Elastic foundation.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): HP (S)
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Hydrostatic pressure on the element reference surface and linear in global *Z*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element reference surface. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Nonuniform pressure on the element reference surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): SP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-4}T^2$
Description: Stagnation pressure applied to the element reference surface.

Load ID (*DSLOAD): ST
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Surface tension applied to the element reference surface.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DSLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element reference surface.

Load ID (*DSLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): VP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-3}T$
Description: Viscous surface pressure applied to the element reference surface. The pressure is proportional to the velocity normal to the element surface and opposing the motion.

Incident Wave Loading

Surface-based incident wave loads are available. They are specified as described in *Acoustic and Shock Loads*. If the incident wave field includes a reflection off a plane outside the boundaries of the mesh, this effect can be included.

Element Output

If a local orientation (*Orientations*) is not used with the element, the stress/strain components are in the default directions on the surface defined by the convention given in *Conventions*. If a local orientation is used with the element, the stress/strain components are in the surface directions defined by the orientation. In large-displacement problems the local directions defined in the reference configuration are rotated into the current configuration by the average material rotation. See *State storage* for details.

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11 Local 11 direct stress.

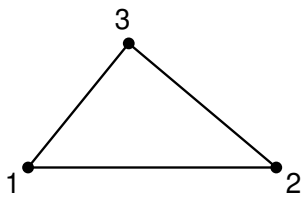
S22 Local 22 direct stress.

S12 Local 12 shear stress.

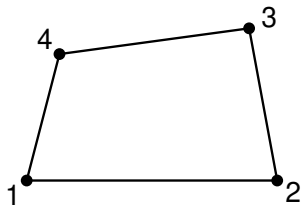
Section Thickness

STH Current thickness.

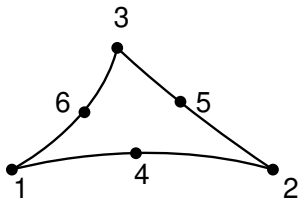
Node Ordering on Elements



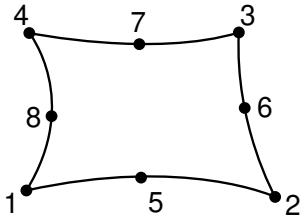
3 - node element



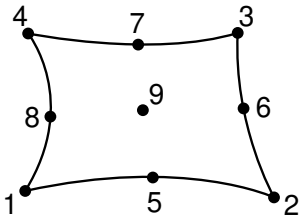
4 - node element



6 - node element

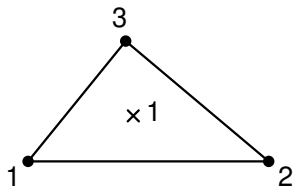


8 - node element

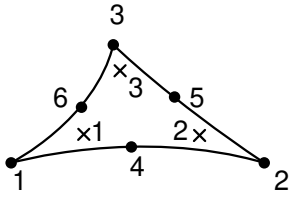


9 - node element

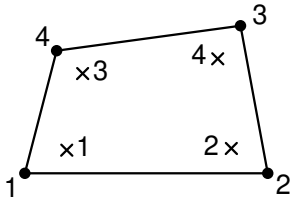
Numbering of Integration Points for Output



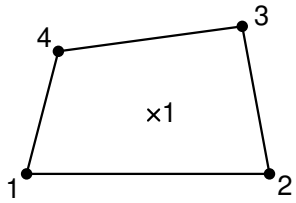
3 - node element



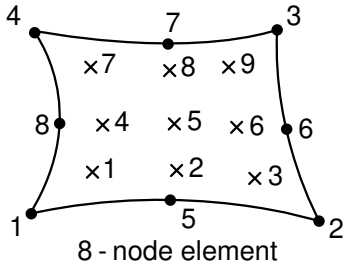
6 - node element



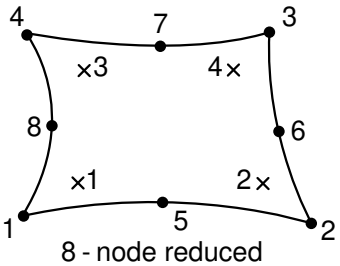
4 - node element



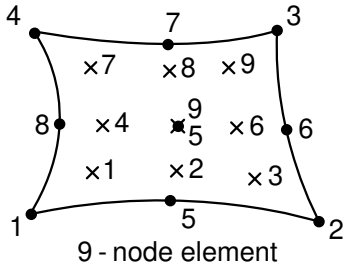
4 - node reduced integration element



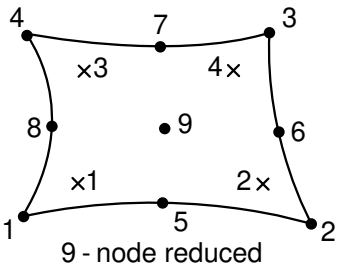
8 - node element



8 - node reduced integration element



9 - node element



9 - node reduced integration element

Cylindrical Membrane Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the cylindrical membrane elements available in Abaqus/Standard.

References:

- *Membrane Elements*
- **MEMBRANE SECTION*

Element Types

MCL6	6-node cylindrical membrane
MCL9	9-node cylindrical membrane

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage: **MEMBRANE SECTION*

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Units: FL^{-3}
Description: Body force in the global *X*-direction.

Load ID (*DLOAD): BY

Units: FL^{-3}

Description: Body force in the global *Y*-direction.

Load ID (*DLOAD): BZ
Units: FL^{-3}

Description: Body force in the global *Z*-direction.

Load ID (*DLOAD): BXNU
Units: FL^{-3}

Description: Nonuniform body force in the global *X*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BYNU
Units: FL^{-3}

Description: Nonuniform body force in the global *Y*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BZNU
Units: FL^{-3}

Description: Nonuniform body force in the global *Z*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): CENT
Units: $FL^{-4}(ML^{-3}\,T^{-2})$

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): CENTRIF
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): CORIO
Units: $FL^{-4}T\ (ML^{-3}\,T^{-1})$

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): GRAV
Units: LT^{-2}

Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): HP
Units: FL^{-2}

Description: Hydrostatic pressure applied to the element reference surface and linear in global *Z*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): P
Units: FL^{-2}

Description: Pressure applied to the element reference surface. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): PNU
Units: FL^{-2}

Description: Nonuniform pressure applied to the element reference surface with magnitude supplied via user subroutine *DLOAD*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): ROTA
Units: T^{-2}
Description: Rotary acceleration load (magnitude is input as α, where α is the rotary acceleration.

Load ID (*DLOAD): ROTDYNF (S)
Units: T^{-1}
Description: Rotordynamic load (magnitude is input as ω, where ω is the angular velocity).

Load ID (*DLOAD): TRSHR
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DLOAD): TRSHRNU (S)
Units: FL^{-2}
Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC
Units: FL^{-2}
Description: General traction on the element reference surface.

Load ID (*DLOAD): TRVECNU (S)
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Foundations

Foundations are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): F
Units: FL^{-3}
Description: Elastic foundation.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): HP
Units: FL^{-2}
Description: Hydrostatic pressure on the element reference surface and linear in global *Z*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): P
Units: FL^{-2}
Description: Pressure on the element reference surface. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): PNU
Units: FL^{-2}

Description: Nonuniform pressure on the element reference surface with magnitude supplied via user subroutine *DLOAD*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): TRSHR
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DSLOAD): TRSHRNU (S)
Units: FL^{-2}
Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Units: FL^{-2}
Description: General traction on the element reference surface.

Load ID (*DSLOAD): TRVECNU (S)
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Element Output

If a local orientation (*Orientations*) is not used with the element, the stress/strain components are expressed in the default directions on the surface defined by the convention given in *Conventions*. If a local orientation is used with the element, the stress/strain components are in the surface directions defined by the orientation. In large-displacement problems the local directions defined in the reference configuration are rotated into the current configuration by the average material rotation. See *State storage* for details.

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11 Local **11** direct stress.

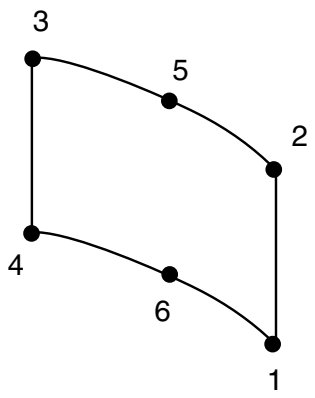
S22 Local **22** direct stress.

S12 Local **12** shear stress.

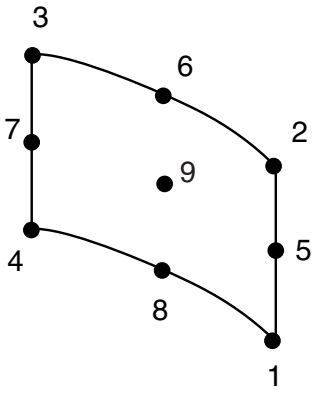
Section Thickness

STH Current thickness.

Node Ordering and Face Numbering on Elements

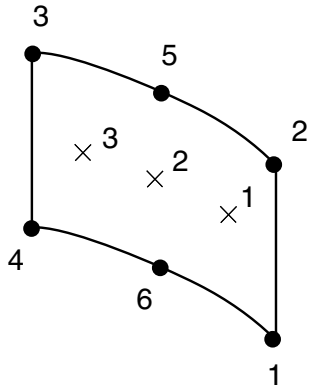


6-node element

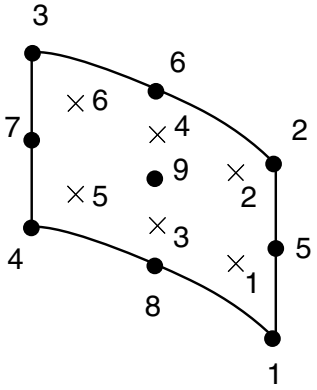


9-node element

Numbering of Integration Points for Output



6-node element



9-node element

Axisymmetric Membrane Element Library

Products: Abaqus/Standard Abaqus/CAE

Overview

This section provides a reference to the axisymmetric membrane elements available in Abaqus/Standard.

Conventions

Coordinate 1 is r, coordinate 2 is z. At $\theta = 0$, the r-direction corresponds to the global X-direction and the z-direction corresponds to the global Y-direction. This is important when data are required in global directions. Coordinate 1 should be greater than or equal to zero.

Degree of freedom 1 is u_r, degree of freedom 2 is u_z. Generalized axisymmetric elements with twist have an additional degree of freedom, 5, corresponding to the twist angle ϕ (in radians).

Abaqus/Standard does not automatically apply any boundary conditions to nodes located along the symmetry axis. You must apply radial or symmetry boundary conditions on these nodes if desired.

Point loads and moments should be given as the value integrated around the circumference; that is, the total value on the ring.

References:

- *Membrane Elements*
- **MEMBRANE SECTION*
- **NODAL THICKNESS*

Element Types

Regular Axisymmetric Membranes

MAX1	2-node linear, without twist
MAX2	3-node quadratic, without twist

Active Degrees of Freedom

1, 2

Additional Solution Variables

None.

Generalized Axisymmetric Membranes

MGAX1	2-node linear, with twist

MGAX2 3-node quadratic, with twist

Active Degrees of Freedom

1, 2, 5

Additional Solution Variables

None.

Nodal Coordinates Required

R, *Z*

Element Property Definition

Input File Usage: **MEMBRANE SECTION*

In addition, use the following option for variable thickness membranes:

**NODAL THICKNESS*

Abaqus/CAE Usage: Property module: **Create Section**: select **Shell** as the section **Category** and **Membrane** as the section **Type**

You cannot define variable thickness membranes in Abaqus/CAE.

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BR
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in the radial (1 or *r*) direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in the axial (2 or *z*) direction.

Load ID (*DLOAD): BRNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in the radial direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in the axial direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): CENT
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-4} ($ML^{-3}T^{-2}$)

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume, ω is the angular velocity). Since only axisymmetric deformation is allowed, the spin axis must be the z-axis.

Load ID (*DLOAD): CENTRIF
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity). Since only axisymmetric deformation is allowed, the spin axis must be the z-axis.

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude input as acceleration).

Load ID (*DLOAD): HP
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Hydrostatic pressure applied to the element reference surface and linear in global Z. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure applied to the element reference surface. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): PNU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure applied to the element reference surface with magnitude supplied via user subroutine *DLOAD*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DLOAD): $TRSHRNU^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC

Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element reference surface.

Load ID (*DLOAD): TRVECNU$^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Foundations

Foundations are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): F
Abaqus/CAE Load/Interaction: Elastic foundation
Units: FL^{-3}
Description: Elastic foundation. For MGAX elements the elastic foundations are applied to degrees of freedom u_r and u_z only.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): HP
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Hydrostatic pressure on the element reference surface and linear in global *Z*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element reference surface. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Nonuniform pressure on the element reference surface with magnitude supplied via user subroutine *DLOAD*. The pressure is positive in the direction opposite of the surface normal.

Load ID (*DSLOAD): ST
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Surface tension on the element reference surface for regular elements.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Shear traction on the element reference surface.

Load ID (*DSLOAD): TRSHRNU(S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element reference surface.

Load ID (*DSLOAD): TRVECNU(S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Incident Wave Loading

Surface-based incident wave loads are available. They are specified as described in *Acoustic and Shock Loads*. If the incident wave field includes a reflection off a plane outside the boundaries of the mesh, this effect can be included.

Element Output

The default local material directions are such that local material direction 1 lies along the line of the element and local material direction 2 is the hoop direction.

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11 Local **11** direct stress.

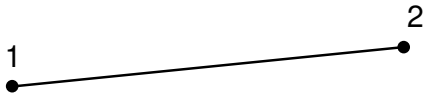
S22 Local **22** direct stress.

S12 Local **12** shear stress. Only available for generalized axisymmetric membrane elements.

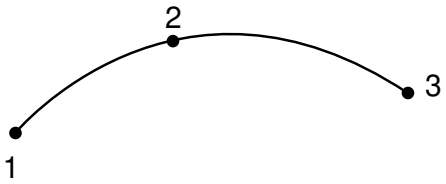
Section Thickness

STH Current thickness.

Node Ordering on Elements

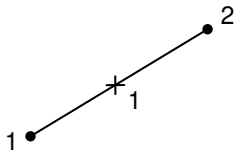


2 - node element

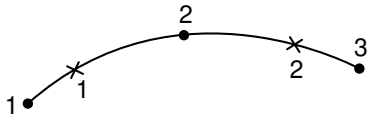


3 - node element

Numbering of Integration Points for Output



2 - node element



3 - node element

Truss Elements

In this section:

- *Truss Elements*
- *Truss Element Library*

Truss Elements

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Truss Element Library*
- **SOLID SECTION*
- *Creating truss sections*

Overview

Truss elements:

- are long, slender structural members that can transmit only axial force (nonstructural link elements are presented in *One-Dimensional Solid (Link) Element Library*); and
- do not transmit moments.

Typical Applications

Truss elements are used in two and three dimensions to model slender, line-like structures that support loading only along the axis or the centerline of the element. No moments or forces perpendicular to the centerline are supported.

The two-dimensional truss elements can be used in axisymmetric models to represent components, such as bolts or connectors, where the strain is computed from the change in length in the r–z plane only. Two-dimensional trusses can also be used to define main surfaces for contact applications in Abaqus/Standard (see *About Contact Interactions*). In this case the direction of the main surface's outward normal is critical for proper detection of contact.

The 3-node truss element available in Abaqus/Standard is often useful for modeling curved reinforcing cables in structures, such as prestressed tendons in reinforced concrete or long slender pipelines used in the off-shore industry.

Choosing an Appropriate Element

A 2-node straight truss element, which uses linear interpolation for position and displacement and has a constant stress, is available in both Abaqus/Standard and Abaqus/Explicit. In addition, a 3-node curved truss element, which uses quadratic interpolation for position and displacement so that the strain varies linearly along the element, is available in Abaqus/Standard.

Hybrid versions of the stress/displacement trusses, coupled temperature-displacement trusses, and piezoelectric trusses are available in Abaqus/Standard.

Hybrid Stress/Displacement Truss Elements

Hybrid (mixed) versions of the stress/displacement trusses, in which the axial force is treated as an additional unknown, are available in two and three dimensions in Abaqus/Standard. These elements are useful (to offset the effects of numerical ill-conditioning on governing equations) when a truss represents a very rigid link whose stiffness is much larger than that of the overall structural model. In such a case a hybrid truss provides an alternative to a truly rigid link, modeled with multi-point constraints (see *General Multi-Point Constraints*) or rigid elements (see *Rigid Elements*).

Coupled Temperature-Displacement Truss Elements

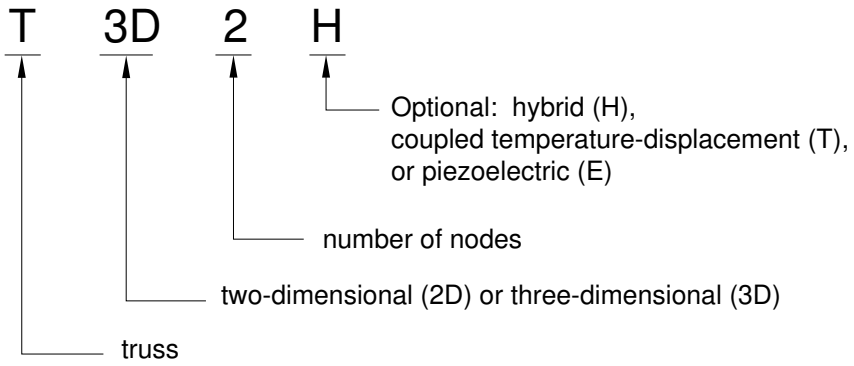
Coupled temperature-displacement truss elements are available in two and three dimensions in Abaqus/Standard. These elements have temperature as an additional degree of freedom (11). See *Fully Coupled Thermal-Stress Analysis* for information about fully coupled temperature-displacement analysis in Abaqus/Standard.

Piezoelectric Truss Elements

Piezoelectric truss elements are available in two and three dimensions in Abaqus/Standard. These elements have electric potential as an additional degree of freedom (9). See *Piezoelectric Analysis* for information about piezoelectric analysis.

Naming Convention

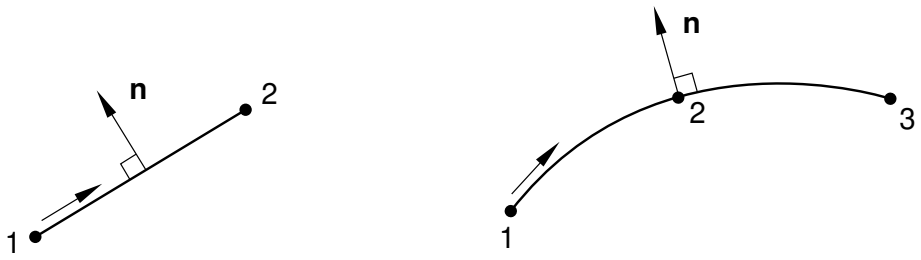
Truss elements in Abaqus are named as follows:



For example, T2D3E is a two-dimensional, 3-node piezoelectric truss element.

Element Normal Definition

For two-dimensional trusses the positive outward normal, $\mathbf{n}$, is defined by a 90° counterclockwise rotation from the direction going from node 1 to node 2 or node 3 of the element, as shown.



Defining the Elements Section Properties

You use a solid section definition to define the section properties. You must associate these properties with a region of your model.

Input File Usage: **SOLID SECTION*, ELSET=*name*

where the ELSET parameter refers to a set of truss elements.

Abaqus/CAE Usage: Property module:
Create Section: select **Beam** as the section **Category** and **Truss** as the section **Type**
Assign->Section: select regions

Defining the Cross-Sectional Area of a Truss Element

You can define the cross-sectional area associated with the truss element as part of the section definition. If you do not specify a value for the cross-sectional area, unit area is assumed.

In a large-displacement Abaqus/Standard analysis, Abaqus/Standard uses the actual material definition when updating the cross-sectional area. However, when truss elements are used in a large-displacement Abaqus/Explicit analysis, Abaqus/Explicit calculates the updated cross-sectional area by assuming that the truss is made of an incompressible material, regardless of the actual material definition. This assumption affects cases only where the strains are large. This assumption is adopted in Abaqus/Explicit because the most common applications of trusses at large strains involve yielding metal behavior or rubber elasticity, in which case the material is effectively incompressible. Therefore, a linear elastic truss element does not provide the same force-displacement response as a linear SPRINGA spring element when the axial strain is not infinitesimal.

Input File Usage: **SOLID SECTION*, ELSET=*name*
cross-sectional area

Abaqus/CAE Usage: Property module: **Create Section**: select **Beam** as the section **Category** and **Truss** as the section **Type**: **Cross-sectional area:** *cross-sectional area*

Assigning a Material Definition to a Set of Truss Elements

You must associate a material definition with each solid section definition. No material orientation is permitted with truss elements.

Input File Usage: **SOLID SECTION*, MATERIAL=*name*

Any value given to the ORIENTATION parameter on the **SOLID SECTION* option will be ignored by truss elements.

Abaqus/CAE Usage: Property module: **Create Section**: select **Beam** as the section **Category** and **Truss** as the section **Type**: **Material:** *name*

Using Truss Elements in Large-Displacement Implicit Analysis

Truss elements have no initial stiffness to resist loading perpendicular to their axis. If a stress-free line of trusses is loaded perpendicular to its axis in Abaqus/Standard, numerical singularities and lack of convergence can result. After the first iteration in a large-displacement implicit analysis, stiffness perpendicular to the initial line of the elements develops, sometimes allowing an analysis to overcome numerical problems.

In some cases loading the truss elements along their axis first or including initial tensile stress can overcome these numerical singularities. However, you must choose the magnitude of the loading or initial stress such that the final solution is unaffected.

Truss Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the truss elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *Truss Elements*
- **SOLID SECTION*

Element Types

2D Stress/Displacement Truss Elements

T2D2	2-node linear displacement
T2D2H$^{(S)}$	2-node linear displacement, hybrid
T2D3$^{(S)}$	3-node quadratic displacement
T2D3H$^{(S)}$	3-node quadratic displacement, hybrid

Active Degrees of Freedom

1, 2

Additional Solution Variables

Element type T2D2H has one additional variable and element type T2D3H has two additional variables relating to axial force.

3D Stress/Displacement Truss Elements

T3D2	2-node linear displacement
T3D2H$^{(S)}$	2-node linear displacement, hybrid
T3D3$^{(S)}$	3-node quadratic displacement
T3D3H$^{(S)}$	3-node quadratic displacement, hybrid

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

Element type T3D2H has one additional variable and element type T3D3H has two additional variables relating to axial force.

2D Coupled Temperature-Displacement Truss Elements

T2D2T$^{(S)}$	2-node, linear displacement, linear temperature
T2D3T$^{(S)}$	3-node, quadratic displacement, linear temperature

Active Degrees of Freedom

1, 2 at middle node for T2D3T

1, 2, 11 at all other nodes

Additional Solution Variables

None.

3D Coupled Temperature-Displacement Truss Elements

T3D2T$^{(S)}$	2-node, linear displacement, linear temperature
T3D3T$^{(S)}$	3-node, quadratic displacement, linear temperature

Active Degrees of Freedom

1, 2, 3 at middle node for T3D3T

1, 2, 3, 11 at all other nodes

Additional Solution Variables

None.

2D Piezoelectric Truss Elements

T2D2E$^{(S)}$	2-node, linear displacement, linear electric potential
T2D3E$^{(S)}$	3-node, quadratic displacement, quadratic electric potential

Active Degrees of Freedom

1, 2, 9

Additional Solution Variables

None.

3D Piezoelectric Truss Elements

T3D2E$^{(S)}$	2-node, linear displacement, linear electric potential
T3D3E$^{(S)}$	3-node, quadratic displacement, quadratic electric potential

Active Degrees of Freedom

1, 2, 3, 9

Additional Solution Variables

None.

Nodal Coordinates Required

2D: *X*, *Y*

3D: *X*, *Y*, *Z*

Element Property Definition

You must provide the cross-sectional area of the element. If no area is given, Abaqus assumes unit area.

Input File Usage: **SOLID SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Beam** as the section **Category** and **Truss** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are available for elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *X*-direction.

Load ID (*DLOAD): BY
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}

Description: Body force in global *Y*-direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *Z*-direction. (Only for 3D trusses.)

Load ID (*DLOAD): BXNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *X*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BYNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *Y*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *Z*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. (Only for 3D trusses.)

Load ID (*DLOAD): CENT$^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-4} ($ML^{-3}T^{-2}$)

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): CENTRIF$^{(S)}$
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): CORIO$^{(S)}$
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-4}T$ ($ML^{-3}T^{-1}$)

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): ROTA$^{(S)}$
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}
Description: Rotary acceleration load (magnitude is input as α, where α is the rotary acceleration).

Abaqus/Aqua Loads

Abaqus/Aqua loads are specified as described in *Abaqus/Aqua Analysis*. They are available only for stress/displacement trusses.

Load ID (*CLOAD/ *DLOAD): FDD
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Transverse fluid drag load.

Load ID (*CLOAD/ *DLOAD): FD1
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Fluid drag force on the first end of the truss (node 1).

Load ID (*CLOAD/ *DLOAD): FD2
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Fluid drag force on the second end of the truss (node 2 or node 3).

Load ID (*CLOAD/ *DLOAD): FDT
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Tangential fluid drag load.

Load ID (*CLOAD/ *DLOAD): FI
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Fluid inertia load.

Load ID (*CLOAD/ *DLOAD): FI1
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Fluid inertia force on the first end of the truss (node 1).

Load ID (*CLOAD/ *DLOAD): FI2
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Fluid inertia force on the second end of the truss (node 2 or node 3).

Load ID (*CLOAD/ *DLOAD): PB
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Buoyancy load (with closed end condition).

Load ID (*CLOAD/ *DLOAD): WDD
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Transverse wind drag load.

Load ID (*CLOAD/ *DLOAD): WD1
Abaqus/CAE Load/Interaction: Not supported
Units: F

Description: Wind drag force on the first end of the truss (node 1).

Load ID (*CLOAD/ *DLOAD): WD2
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Wind drag force on the second end of the truss (node 2 or node 3).

Distributed Heat Fluxes

Distributed heat fluxes are available for coupled temperature-displacement trusses. They are specified as described in *Thermal Loads*.

Load ID (*DFLUX): $BF^{(S)}$
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}\,T^{-1}$
Description: Heat body flux per unit volume.

Load ID (*DFLUX): $BFNU^{(S)}$
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}\,T^{-1}$
Description: Nonuniform heat body flux per unit volume with magnitude supplied via user subroutine *DFLUX*.

Load ID (*DFLUX): $S1^{(S)}$
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}\,T^{-1}$
Description: Heat surface flux per unit area into the first end of the truss (node 1).

Load ID (*DFLUX): $S2^{(S)}$
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}\,T^{-1}$
Description: Heat surface flux per unit area into the second end of the truss (node 2 or node 3).

Load ID (*DFLUX): $S1NU^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}\,T^{-1}$
Description: Nonuniform heat surface flux per unit area into the first end of the truss (node 1) with magnitude supplied via user subroutine *DFLUX*.

Load ID (*DFLUX): $S2NU^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}\,T^{-1}$
Description: Nonuniform heat surface flux per unit area into the second end of the truss (node 2 or node 3) with magnitude supplied via user subroutine *DFLUX*.

Film Conditions

Film conditions are available for coupled temperature-displacement trusses. They are specified as described in *Thermal Loads*.

Load ID (*FILM): $F1^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}\,T^{-1}\theta^{-1}$

Description: Film coefficient and sink temperature at the first end of the truss (node 1).

Load ID (*FILM): F2$^{(S)}$

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}\,T^{-1}\theta^{-1}$

Description: Film coefficient and sink temperature at the second end of the truss (node 2 or node 3).

Load ID (*FILM): F1NU$^{(S)}$

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}\,T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature at the first end of the truss (node 1) with magnitude supplied via user subroutine *FILM*.

Load ID (*FILM): F2NU$^{(S)}$

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}\,T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature at the second end of the truss (node 2 or node 3) with magnitude supplied via user subroutine *FILM*.

Radiation Types

Radiation conditions are available for coupled temperature-displacement trusses. They are specified as described in *Thermal Loads*.

Load ID (*RADIATE): R1$^{(S)}$

Abaqus/CAE Load/Interaction: Surface radiation

Units: Dimensionless

Description: Emissivity and sink temperature at the first end of the truss (node 1).

Load ID (*RADIATE): R2$^{(S)}$

Abaqus/CAE Load/Interaction: Surface radiation

Units: Dimensionless

Description: Emissivity and sink temperature at the second end of the truss (node 2 or node 3).

Electric Fluxes

Electric fluxes are available for piezoelectric trusses. They are specified as described in *Piezoelectric Analysis*.

Load ID (*DECHARGE): EBF$^{(S)}$

Abaqus/CAE Load/Interaction: Body charge

Units: CL^{-3}

Description: Body flux per unit volume.

Element Output

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11

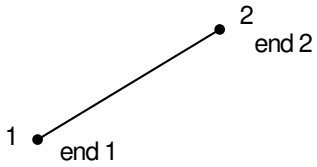
Axial stress.

Heat Flux Components

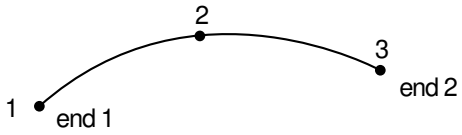
Available for coupled temperature-displacement trusses.

HFL1 Heat flux along the element axis.

Node Ordering on Elements

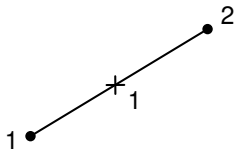


2 - node element

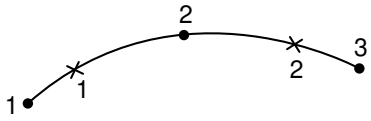


3 - node element

Numbering of Integration Points for Output



2 - node element



3 - node element

Beam Elements

In this section:

- *About Beam Modeling*
- *Choosing a Beam Cross-Section*
- *Choosing a Beam Element*
- *Beam Element Cross-Section Orientation*
- *Beam Section Behavior*
- *Using a Beam Section Integrated during the Analysis to Define the Section Behavior*
- *Using a General Beam Section to Define the Section Behavior*
- *Beam Element Library*
- *Beam Cross-Section Library*

About Beam Modeling

Overview

Beam modeling consists of:

- choosing a beam cross-section (*Choosing a Beam Cross-Section* and *Beam Cross-Section Library*);
- choosing the appropriate beam element type (*Choosing a Beam Element* and *Beam Element Library*);
- defining the beam cross-section orientation (*Beam Element Cross-Section Orientation*);
- determining whether or not numerical integration is needed to define the beam section behavior (*Beam Section Behavior*); and
- defining the beam section behavior (*Using a Beam Section Integrated during the Analysis to Define the Section Behavior* or *Using a General Beam Section to Define the Section Behavior*).

Abaqus offers a wide range of beam modeling options.

References:

Determining Whether Beam Modeling Is Appropriate

Beam theory is the one-dimensional approximation of a three-dimensional continuum. The reduction in dimensionality is a direct result of slenderness assumptions; that is, the dimensions of the cross-section are small compared to typical dimensions along the axis of the beam. The axial dimension must be interpreted as a global dimension (not the element length), such as

- distance between supports,
- distance between gross changes in cross-section, or
- wavelength of the highest vibration mode of interest.

In Abaqus a beam element is a one-dimensional line element in three-dimensional space or in the X–Y plane that has stiffness associated with deformation of the line (the beam's “axis”). These deformations consist of axial stretch; curvature change (bending); and, in space, torsion. (*Truss Elements* are one-dimensional line elements that have only axial stiffness.) Beam elements offer additional flexibility associated with transverse shear deformation between the beam's axis and its cross-section directions. Some beam elements in Abaqus/Standard also include warping—nonuniform out-of-plane deformation of the beam's cross-section—as a nodal variable. The main advantage of beam elements is that they are geometrically simple and have few degrees of freedom. This simplicity is achieved by assuming that the member's deformation can be estimated entirely from variables that are functions of position along the beam axis only. Thus, a key issue in using beam elements is to judge whether such one-dimensional modeling is appropriate.

The fundamental assumption used is that the beam section (the intersection of the beam with a plane that is perpendicular to the beam axis; see the discussion in *Choosing a Beam Cross-Section*) cannot deform in its own plane (except for a constant change in cross-sectional area, which may be introduced in geometrically nonlinear analysis and causes a strain that is the same in all directions in the plane of the section). The implications of this assumption should be considered carefully in any use of beam elements, especially for cases involving large amounts of bending or axial tension/compression of non-solid cross-sections such as pipes, I-beams, and U-beams. Section collapse may occur and result in very weak behavior that is not predicted by beam theory. Similarly, thin-walled, curved pipes exhibit much softer bending behavior than would be predicted by beam theory because

the pipe wall readily bends in its own section—another effect precluded by this basic assumption of beam theory. This effect, which must generally be considered when designing piping elbows, can be modeled by using shell elements to model the pipe as a three-dimensional shell (see *About Shell Elements*) or, in Abaqus/Standard, by using elbow elements (see *Pipes and Pipebends with Deforming Cross-Sections: Elbow Elements*).

In addition to beam elements, frame elements are provided in Abaqus/Standard. These elements provide efficient modeling for design calculations of frame-like structures composed of initially straight, slender members. They operate directly in terms of axial force, bending moments, and torque at the element's end nodes. They are implemented for small or large displacements (large rotations with small strains) and permit the formation of plastic hinges at their ends through a “lumped” plasticity model that includes kinematic hardening. See *Frame Elements* for details.

In addition to the various beam elements, Abaqus also provides pipe elements to model beams with pipe cross-sections that are subject to internal stress due to internal and/or external pressure loading. Abaqus provides a choice of two formulations for pipe elements:

- the thin-walled formulation, where the hoop stress is assumed to be constant and the radial stress is neglected, is available in Abaqus/Explicit and Abaqus/Standard; and
- the thick-walled formulation, where the hoop and radial stress vary through the cross-section, is available only in Abaqus/Standard.

The pipe elements are a specialized form of the corresponding beam elements that allow for internal and/or external pressure load specification and take the resulting hoop stress (as well as radial stress for thick-walled pipes) into account for the material constitutive calculations. Usage of the pipe elements is identical to that of the corresponding beam elements with respect to the section definition, boundary conditions at the element nodes, surface definitions, interactions such as tie constraints, etc.

Using Beam Elements in Dynamic and Eigenfrequency Analysis

The rotary inertia of a beam cross-section is usually insignificant for slender beam structures, except for twist around the beam axis. Therefore, Abaqus/Standard ignores rotary inertia of the cross-section for Euler-Bernoulli beam elements in bending. For thicker beams the rotary inertia plays a role in dynamic analysis, but to a lesser extent than shear deformation effects.

For Timoshenko beams the inertia properties are calculated from the cross-section geometry. The rotary inertia associated with torsional modes is different from that of flexural modes. For unsymmetric cross-sections the rotary inertia is different in each direction of bending. Abaqus allows you to choose the rotary inertia formulation for Timoshenko beams. When an approximate isotropic formulation is requested, the rotary inertia associated with the torsional mode is used for all rotational degrees of freedom in Abaqus/Standard, and a scaled flexural inertia with a scaling factor chosen to maximize the stable time increment is used for all rotational degrees of freedom in Abaqus/Explicit. The center of mass of the cross-section is taken to be located at the beam node. When the exact (anisotropic) formulation is requested, the rotary inertia associated with bending and torsion differ and the coupling between the translational and rotational degrees of freedom is included for beam cross-section definitions where the beam node is not located at the center of mass of the cross-section. For Timoshenko beams with the exact (default) rotary inertia formulation, you can define an additional mass and rotary inertia contribution to the beam's inertia response that does not add to its structural stiffness; see *Adding Inertia to the Beam Section Behavior for Timoshenko Beams*.

Choosing a Beam Cross-Section

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Beam Modeling*
- *Beam Cross-Section Library*
- *Meshed Beam Cross-Sections*
- *Defining profiles*

Overview

The choice of cross-section is determined by the geometry of the cross-section and its behavior. A beam's cross-section:

- can be solid or thin-walled;
- if thin-walled, can be open or closed; and
- can be defined by choosing from the Abaqus cross-section library; by specifying geometric quantities such as area, moments of inertia, and torsional constant; or by using a mesh of special two-dimensional elements, for which geometric quantities are calculated numerically.

You must consider whether the section should be treated as a solid cross-section or as a thin-walled cross-section. This choice determines the basis upon which Abaqus computes the axial and shear strains at each point in the section.

Solid Cross-Sections

For solid sections under bending, plane (beam) sections remain plane. Under torsional loading any noncircular beam section will warp: the beam section will not remain planar. However, for solid sections the warping of the section is small enough so that the axial strain due to warping of the section can be neglected and St. Venant warping theory can be used to construct a single component of shear strain at each integration point in the section. This is done automatically for the rectangular and trapezoidal sections in the beam section library. The St. Venant warping functions are used to define the shear strain even when the response in the section is no longer purely elastic. This limits the accuracy of the modeling for cases involving noncircular solid beam sections subjected to torsional loadings that cause large amounts of inelastic deformation. When using a meshed beam profile, two shear strain components are available for output in the user-specified material system. The thick pipe section is treated as a solid cross-section.

Nonsolid (“Thin-Walled”) Cross-Sections

In Abaqus nonsolid sections are treated as “thin-walled” sections; that is, in the plane of the section, the thickness of a branch of the section is assumed to be small compared to its length. Thin-walled beam theory determines the shear in the wall of the section depending on whether the section is closed or open.

Closed Sections

A closed section is a nonsolid section whose branches form closed loops. Closed sections offer significant resistance to torsion and do not warp significantly. Abaqus ignores warping effects for closed sections.

In Abaqus predefined beam sections can model only one closed loop. Sections with multiple loops must be modeled with a meshed beam section (see *Meshed Cross-Sections*) or with shells.

For sufficiently small thickness of the section walls, the variation of shear stress across the thickness is negligible; the formulation of the closed sections available in Abaqus is based on this assumption.

Open Sections

An open section is a nonsolid section with branches that do not form closed loops, such as an I-section or a U-section. In such sections the shear stress is assumed to vary linearly over the wall thickness and to vanish at the center of the wall. Open sections can warp significantly and generally require the use of open-section warping theory (available with beam element types B*xx*OS in Abaqus/Standard) with suitable warping constraints (applied to degree of freedom 7) at supports or joints. Such warping constraints may significantly increase the torsional stiffness of the beam. Open, thin-walled sections whose branches are straight lines that meet at a single point (such as the L-section in the Abaqus beam element section library, T-sections, or X-sections) do not warp; therefore, warping constraints have no effect. Such sections always have very little torsional stiffness.

If an open section is used with a regular beam element type (not B*xx*OS), the open section is assumed to be free to warp and the axial strain due to warping is neglected. Consequently, the section will have very little torsional stiffness.

Section Property Calculations

Thin-walled assumptions are used when calculating nonsolid section properties. Properties for sections comprised of intersecting straight segments (arbitrary, box, channel, hat, hexagonal, I-, and L-sections) also include an approximation of the intersection geometry.

Available Beam Cross-Sections

You can specify any of the following types of beam cross-sections: an Abaqus library cross-section, a generalized cross-section for which you specify the geometric quantities directly, or a meshed cross-section.

Abaqus Beam Cross-Section Library

The Abaqus beam cross-section library contains

- solid sections (circular, rectangular, and trapezoidal);
- closed thin-walled sections (box, hexagonal, and pipe);
- open thin-walled sections (channel-shaped, hat-shaped, I-shaped, L-shaped, and T-shaped); and
- a thick-walled pipe section.

Abaqus also provides an arbitrary thin-walled section definition; Abaqus treats this section type as a closed or open section, depending on how the section is defined.

Arbitrary, channel, hat, I, and trapezoidal library sections allow you to define the location of the origin of the local coordinate system. Other section types (such as rectangular, circular, L, or pipe) have preset origins. The channel and hat sections are available only with general beam section definitions.

Input File Usage: Use the following option to define a beam section integrated during the analysis:

**BEAM SECTION*, SECTION=*name*

where *name* can be ARBITRARY, BOX, CIRC, HEX, I, L, PIPE, RECT, THICK PIPE, or TRAPEZOID. A T-section is defined by specifying geometric data for only one flange of an I-section.

Use the following option to define a general beam section:

**BEAM GENERAL SECTION*, SECTION=*name*

where *name* can be ARBITRARY, BOX, CHANNEL, CIRC, HAT, HEX, I, L, PIPE, RECT, or TRAPEZOID. A T-section is defined by specifying geometric data for only one flange of an I-section.

Abaqus/CAE Usage: Property module: **Create Profile**: choose **Box**, **Pipe**, **Circular**, **Rectangular**, **Hexagonal**, **Trapezoidal**, **I**, **L**, **T**, or **Arbitrary**

The channel and hat beam cross-sections are not supported in Abaqus/CAE.

Generalized Cross-Sections

Abaqus also allows you to specify "generalized" cross-sections by specifying the geometric quantities necessary to define the section. Such generalized sections can be used only with linear material behavior although the section response can be linear or nonlinear.

Input File Usage: Use the following option to define a linear generalized cross-section:

**BEAM GENERAL SECTION*, SECTION=GENERAL

Use the following option to define a nonlinear generalized cross-section:

**BEAM GENERAL SECTION*, SECTION=NONLINEAR GENERAL

Abaqus/CAE Usage: Property module: **Create Profile**: choose **Generalized**

Nonlinear generalized cross-sections are not supported in Abaqus/CAE.

Meshed Cross-Sections

Abaqus allows you to mesh an arbitrarily shaped solid cross-section by using warping elements (see *Warping Elements*) in a two-dimensional analysis to generate beam cross-section properties that can be used in a subsequent two- or three-dimensional beam analysis. Such sections permit only linear, elastic material behavior. Therefore, a meshed cross-section can be used only with a general beam section definition; for details, see *Meshed Beam Cross-Sections*.

Input File Usage: **BEAM GENERAL SECTION*, SECTION=MESHED

Abaqus/CAE Usage: Meshed cross-sections are not supported in Abaqus/CAE.

Choosing a Beam Element

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Beam Modeling*
- *Beam Element Library*
- **TRANSVERSE SHEAR STIFFNESS*
- **SECTION STIFFNESS*
- *Creating beam sections*

Overview

Abaqus offers a wide range of beam elements, including “Euler-Bernoulli”-type beams and “Timoshenko”-type beams with solid, thin-walled closed and thin-walled open sections.

The Abaqus/Standard beam element library includes:

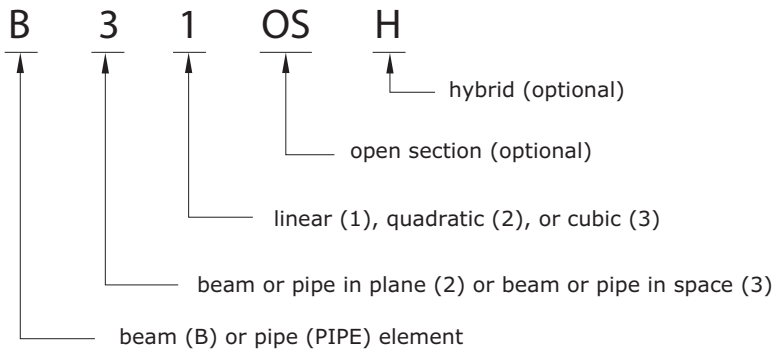
- Euler-Bernoulli (slender) beams in a plane and in space;
- Timoshenko (shear flexible) beams in a plane and in space;
- linear, quadratic, and cubic interpolation formulations;
- warping (open section) beams;
- pipe elements; and
- hybrid formulation beams, typically used for very stiff beams that rotate significantly (applications in robotics or in very flexible structures such as offshore pipelines).

The Abaqus/Explicit beam element library includes:

- Timoshenko (shear flexible) beams in a plane and in space;
- linear and quadratic interpolation formulations; and
- linear pipe elements.

Naming Convention

Beam elements in Abaqus are named as follows:



For example, B21H is a planar beam that uses linear interpolation and a hybrid formulation.

Euler-Bernoulli (Slender) Beams

Euler-Bernoulli beams (B23, B23H, B33, and B33H) are available only in Abaqus/Standard. These elements do not allow for transverse shear deformation; plane sections initially normal to the beam's axis remain plane (if there is no warping) and normal to the beam axis. They should be used only to model slender beams: the beam's cross-sectional dimensions should be small compared to typical distances along its axis (such as the distance between support points or the wavelength of the highest mode that participates in a dynamic response). For beams made of uniform material, typical dimensions in the cross-section should be less than about 1/15 of typical axial distances for transverse shear flexibility to be negligible. (The ratio of cross-section dimension to typical axial distance is called the slenderness ratio.)

Load stiffness for pressure loads is not included for these elements.

Interpolation

The Euler-Bernoulli beam elements use cubic interpolation functions, which makes them reasonably accurate for cases involving distributed loading along the beam. Therefore, they are well suited for dynamic vibration studies, where the d'Alembert (inertia) forces provide such distributed loading.

The cubic beam elements are written for small-strain, large-rotation analysis. They may not be appropriate for torsional stability problems due to the approximations in the underlying formulation and cannot be used in analyses involving very large rotations (of the order 180°); quadratic or linear beam elements should be used instead.

Mass Formulation

The Euler-Bernoulli beam elements use a consistent mass formulation. Rotary inertia for twist around the beam axis is the same as for Timoshenko beams. For details, see *Mass and inertia for Timoshenko beams*. Any additional inertia defined for these elements (see *Adding Inertia to the Beam Section Behavior for Timoshenko Beams*) is ignored.

Timoshenko (Shear Flexible) Beams

Timoshenko beams (B21, B22, B31, B31OS, B32, B32OS, PIPE21, PIPE22, PIPE31, PIPE32, and their “hybrid” equivalents) allow for transverse shear deformation. They can be used for thick (“stout”) as well as slender beams. For beams made from uniform material, shear flexible beam theory can provide useful results for cross-sectional dimensions up to 1/8 of typical axial distances or the wavelength of the highest natural mode that contributes significantly to the response. Beyond this ratio the approximations that allow the member's behavior to be described solely as a function of axial position no longer provide adequate accuracy.

By default, Abaqus assumes that the transverse shear behavior of Timoshenko beams is linear elastic with a fixed modulus and, therefore, independent of the response of the beam section to axial stretch and bending. In Abaqus/Standard, B31 and B32 elements that reference meshed cross-sections might also have full coupling between the section strains (axial stretch, transverse shear, torsion, and bending). Elements with full coupling can accurately characterize the response of beams that are made out of composite materials, such as wind turbine rotor blades. The response of such elements is still linear elastic, but the transverse shear behavior and axial stretch, torsion, and bending cannot be separated (see *Defining the Stiffness and Mass Properties for Meshed General Beam Sections* below).

For most beam sections Abaqus calculates the transverse shear stiffness values required in the element formulation. You can override these default values as described below in *Defining the Transverse Shear Stiffness and the Slenderness Compensation Factor*. The default shear stiffness values are not calculated in some cases if estimates of shear moduli are unavailable during the preprocessing stage of input; for example, when the material behavior is defined by user subroutine *UMAT*, *UHYPEL*, *UHYPER*, or *VUMAT*. In such cases you must define the transverse

shear modulus for the material (see *Defining the Elastic Transverse Shear Modulus*) or the transverse shear stiffness for the section as described below.

The Timoshenko beams can be subjected to large axial strains. The axial strains due to torsion are assumed to be small. In combined axial-torsion loading, torsional shear strains are calculated accurately only when the axial strain is not large.

Defining the Stiffness and Mass Properties for Meshed General Beam Sections

You can define the stiffness and mass properties for meshed general beam sections with or without full coupling among section strains. Without full coupling, the transverse shear behavior of Timoshenko beams is independent of the response of the beam section to axial stretch and bending, as is the case for other section types. Therefore, you define the transverse shear stiffness behavior separately. With full coupling among section strains, you define 21 values of a symmetric 6×6 stiffness matrix that relate the six-component section strain vector to the six-components' section forces and moments vector. This definition includes the transverse shear behavior. In both cases, you define the mass properties separately.

Typically, you do not define the stiffness and mass properties for meshed general beam sections manually. Instead, you may define and analyze a two-dimensional mesh of the cross-section geometry (optionally with 3-DOF warping elements to capture all potential coupling effects) with proper material assignments in Abaqus/Standard. Such a preliminary analysis generates a *jobname*`.bsp` file that contains the options to define the section stiffness and mass properties and that you can include to associate these definitions with a general beam section (see *Meshed Beam Cross-Sections*).

Input File Usage:

Use the following options to define the section stiffness and mass properties for meshed general beam sections without full coupling (to define the transverse shear behavior, see *Defining the Transverse Shear Stiffness and the Slenderness Compensation Factor* below):

**BEAM GENERAL SECTION*, SECTION=`MESHED`
**SECTION STIFFNESS*

(EA), $(EI)_{11}$, $(EI)_{12}$, $(EI)_{22}$, (GJ)
**SECTION INERTIA*

(ρA), $(\rho I)_{11}$, $(\rho I)_{12}$, $(\rho I)_{22}$, x_{1cm}, x_{2cm}

Use the following options to define the section stiffness and mass properties for meshed general beam sections with full coupling between the section strains, including the transverse shear behavior:

**BEAM GENERAL SECTION*, SECTION=`MESHED`
**SECTION STIFFNESS*, FULL COUPLING

$(K)_{11}$, $(K)_{22}$, $(K)_{33}$, $(K)_{44}$, $(K)_{55}$, $(K)_{66}$, $(K)_{12}$, $(K)_{13}$
$(K)_{14}$, $(K)_{15}$, $(K)_{16}$, $(K)_{23}$, $(K)_{24}$, $(K)_{25}$, $(K)_{26}$, $(K)_{34}$
$(K)_{35}$, $(K)_{36}$, $(K)_{45}$, $(K)_{46}$, $(K)_{56}$
**SECTION INERTIA*

(ρA), $(\rho I)_{11}$, $(\rho I)_{12}$, $(\rho I)_{22}$, x_{1cm}, x_{2cm}

Use the following options to associate the meshed beam general section definition with the stiffness and mass properties that you computed with a beam section generate procedure:

**BEAM GENERAL SECTION*, SECTION=`MESHED`
**INCLUDE*, INPUT=*jobname*`.bsp`

Abaqus/CAE Usage: Defining meshed general beam sections is not supported in Abaqus/CAE.

Transverse Shear Stiffness Definition

The effective transverse shear stiffness of the section of a shear flexible beam is defined in Abaqus as

$$\overline{K}_{\alpha 3} = f_p^{\alpha} K_{\alpha 3},$$

where $\overline{K}_{\alpha 3}$ is the section shear stiffness in the α-direction; f_p^{α} is a dimensionless factor used to prevent the shear stiffness from becoming too large in slender beam elements; $K_{\alpha 3}$ is the actual shear stiffness of the section; and $\alpha = 1, 2$ are the local directions of the cross-section. The $K_{\alpha 3}$ have units of force.

The dimensionless factors f_p^{α} are always included in the calculation of transverse shear stiffness and are defined as

$$f_p^{\alpha} = 1/\left(1 + \xi * SCF\frac{l^2 A}{12 I_{\alpha\alpha}}\right),$$

where l is the length of the element, A is the cross-sectional area, $I_{\alpha\alpha}$ is the inertia in the α-direction, SCF is the slenderness compensation factor (with a default value of 0.25), and ξ is a constant of value 1.0 for first-order elements and 10^{-4} for second-order elements.

For meshed cross-sections the above expressions change to

$$\overline{K}_{\alpha\beta}^{ts} = f_p K_{\alpha\beta}^{ts},$$

$$f_p = 1/\left(1 + \xi * SCF\frac{l^2 (EA)}{12 (EI)_{av}}\right),$$

$$(EI)_{av} = 1/3\left((EI)_{11} + (EI)_{22} + (EI)_{12}\right).$$

If the meshed cross-sections exhibit full coupling between the section strains, no slenderness compensation is applied; that is, $f_p = 1$.

You can define the $K_{\alpha 3}$ or $K_{\alpha\beta}^{ts}$ as described below. If you do not specify them, they are defined by

$$K_{\alpha 3} = kGA,$$

or

$$K_{\alpha\beta}^{ts} = k(GA)_{\alpha\beta},$$

where G is the elastic shear modulus or moduli and A is the cross-sectional area of the beam section. Abaqus calculates the transverse shear stiffness, $K_{\alpha 3}$ or $K_{\alpha\beta}^{ts}$, only once at the beginning of the analysis based on the initial value of G. Initial temperature and field variable dependencies of G are considered. Any changes to the transverse shear stiffness that occur due to changes in the material stiffness during the analysis are ignored. The shear factor k (Cowper, 1966) is defined as:

Section type	Shear factor, k
Arbitrary	1.0
Box	0.44
Channel	1.0
Circular	0.89
Elbow	0.85

Section type	Shear factor, k
Generalized	1.0
Hat	1.0
Hexagonal	0.53
I (and T)	0.44
L	1.0
Meshed	1.0
Nonlinear generalized	1.0
Pipe	0.53
Rectangular	0.85
Thick pipe	0.53–0.89
Trapezoidal	0.822

When a beam section definition integrated during the analysis is used (see *Using a Beam Section Integrated during the Analysis to Define the Section Behavior*) or when a general beam section definition is used with a material definition (see *Using a General Beam Section to Define the Section Behavior*), G is calculated from the elastic material definition used with the section. When a general beam section definition is used without a material definition (see *Using a General Beam Section to Define the Section Behavior*), you provide G as part of the beam section data.

Defining the Transverse Shear Stiffness and the Slenderness Compensation Factor

You can define the transverse shear stiffness for beam sections integrated during the analysis and general beam sections. In the case of two-dimensional beams, you can input a single value of transverse shear stiffness (that is, K_{23}). For meshed cross-sections, you define the shear stiffnesses in the first and second directions (K_{11}^{ts} and K_{22}^{ts}, respectively) and a coupling term for the two directions (K_{12}^{ts}). In the case of full coupling between the section strains, you define the section stiffness and specify the transverse shear stiffness, along with the stiffness for axial stretch, torsion, and bending and the coupling stiffnesses (see *Defining the Stiffness and Mass Properties for Meshed General Beam Sections*).

When you define the transverse shear stiffness, you can also define the slenderness compensation factor. The default value for the slenderness compensation factor is 0.25. If you provide a slenderness compensation factor value, you must also provide the values of the shear stiffness $K_{\alpha 3}$.

In the case of first-order elements, you can define the slenderness compensation factor by including the label SCF. Abaqus then uses a slenderness compensation factor of $SCF = \frac{kGA}{EA}$, and any values of $K_{\alpha 3}$ that you specify are ignored. Instead, the $K_{\alpha 3}$ values are calculated from the elastic material definition.

The transverse shear stiffness is not relevant to Euler-Bernoulli beam elements for which the transverse shear constraints are satisfied exactly.

Input File Usage: Use both of the following options to define the transverse shear stiffness for beam sections integrated during the analysis:

**BEAM SECTION*
**TRANSVERSE SHEAR STIFFNESS*

Use both of the following options to define the transverse shear stiffness for general beam sections:

**BEAM GENERAL SECTION*
**TRANSVERSE SHEAR STIFFNESS*

Abaqus/CAE Usage: To define transverse shear stiffness for beam sections integrated during the analysis:

Property module: beam section editor: **Section integration: During analysis**: **Stiffness**: toggle on **Specify transverse shear**

To define transverse shear stiffness for general beam sections:

Property module: beam section editor: **Section integration: Before analysis**: **Stiffness**, toggle on **Specify transverse shear**

Interpolation

Abaqus provides finite axial strain, shear flexible beams with linear and quadratic interpolations. Their formulation is described in *Beam element formulation*.

Element types B21, B31, B31OS, PIPE21, PIPE31, and their hybrid equivalents use linear interpolation. These elements are well suited for cases involving contact, such as the laying of a pipeline in a trench or on the seabed or the contact between a drill string and a well hole, and for dynamic versions of similar problems (impact).

Element types B22, B32, B32OS, PIPE22, PIPE32, and their hybrid equivalents use quadratic interpolation.

Mass Formulation

The linear Timoshenko beam elements use a lumped mass formulation by default. The quadratic Timoshenko beam elements in Abaqus/Standard use a consistent mass formulation, except in dynamic procedures in which a lumped mass formulation with a 1/6, 2/3, 1/6 distribution is used. For details, see *Mass and inertia for Timoshenko beams*. The quadratic Timoshenko beam elements in Abaqus/Explicit use a lumped mass formulation with a 1/6, 2/3, 1/6 distribution.

Using a Consistent Mass Matrix in Abaqus/Standard

Alternatively, in Abaqus/Standard you can use the McCalley-Archer consistent mass matrix based on the cubic interpolation of deflections and quadratic interpolation of rotations.

Input File Usage: Use the following option for linear Timoshenko beam elements with beam sections integrated during the analysis:

**BEAM SECTION*, LUMPED=NO

Use the following option for linear Timoshenko beam elements with general beam sections:

**BEAM GENERAL SECTION*, LUMPED=NO

Abaqus/CAE Usage: Use the following option for linear Timoshenko beam elements with beam sections integrated during the analysis:

Property module: beam section editor: **Section integration: During analysis**: **Stiffness** tabbed page: toggle on **Use consistent mass matrix formulation**

Use the following option for linear Timoshenko beam elements with general beam sections:

Property module: beam section editor: **Section integration: Before analysis**: **Stiffness** tabbed page: toggle on **Use consistent mass matrix formulation**

Rotary Inertia Treatment and Additional Beam Inertia

By default, the exact (anisotropic with displacement-rotation coupling) rotary inertia is used for Timoshenko beams. Optionally, an uncoupled isotropic approximation to the rotary inertia can be used. See *Rotary Inertia for Timoshenko Beams* for further details.

The exception to this rule is the static procedure with automatic stabilization (see *Static Stress Analysis*), where the mass matrix for Timoshenko beams is always calculated assuming isotropic rotary inertia, regardless of the type of rotary inertia specified for the beam section definition (see *Rotary Inertia for Timoshenko Beams*).

In some structural applications the beam element may be a one-dimensional approximation of a structure with complex cross-sectional geometry and mass distribution. In such a cross-section there may be inertia contributions that represent heavy machinery, cargo loaded in a ship compartment, fluid-filled ballast tanks, or any other mass distributed along the length of the beam that is not part of the beam's structural stiffness. In such cases you can define additional mass and rotary inertia associated with the beam section properties. Multiple masses per unit length (with location other than the origin of the beam cross-section) and rotary inertias per unit length can be specified. Mass proportional damping (alpha or composite damping) associated with this additional inertia can also be specified. Abaqus uses the mass weighted average (based on the material damping and the added inertial damping) for the element mass proportional damping. See *Material Damping* for details.

Additional Inertia due to Immersion in Fluid

When a beam is fully or partially submerged, the effect of the surrounding fluid can be modeled as an additional distributed inertia on the beam. See *Additional Inertia due to Immersion in Fluid* for details.

Warping (Open-Section) Beams

When modeling beams in space, a further consideration arises from the possible warping of the beam's cross-section under torsional loading. For all but circular sections the beam's cross-section deforms out of its original plane when subject to torsion. This warping deformation modifies the shear strain distribution throughout the section.

Open sections typically twist very easily if warping is not prevented, especially if the walls that form the beam section are thin. Constraint of this warping at certain points along the beam (such as where the beam is built into some other member, *Figure 1*, or into a wall) is then a major determinant of the beam's overall torsional response.

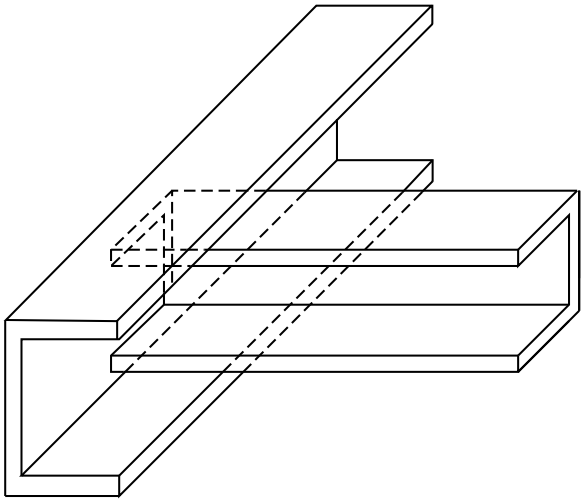


Figure 1: Intersection of open section beams.

Element types B31OS, B32OS (and their “hybrid” equivalents) have the warping magnitude, w, as a degree of freedom at each node; they are available only in Abaqus/Standard. In these elements Abaqus/Standard assumes that the warping of the cross-section follows a certain pattern as a function of position in the cross-section (Abaqus calculates this warping pattern if you have specified a standard library section or an “arbitrary” section): only the warping magnitude varies with position along the beam's axis. These elements are meant for the analysis of thin-walled open sections in which warping constraints play a role and the axial strains due to warping cannot be neglected. Examples of such open sections that may warp in this fashion are the I-section, channel section, hat section, and any open arbitrary section. In the other beam element types warping is considered unconstrained and any axial stress due to warping is neglected; torsional behavior is not represented adequately when these element types are used with thin-walled, open sections.

In general, the warping magnitude can be continuous only when the beam axis is continuous through a node and the beam cross-section is the same on both sides of the node. Thus, if open-section members intersect at a node (such as the cross-member of a vehicle chassis abutting a longitudinal member, *Figure 1*), separate nodes may have to be used for the intersecting members with different axial directions and appropriate constraints must be chosen for the warping amplitudes in each member at this point. The choice of these constraints is a matter of detail of the local construction. For example, if the joint is reinforced, warping may be prevented; therefore, degree of freedom 7 should be fully constrained with a boundary condition on the appropriate members at the joint.

“Pipe” Elements

The pipe elements in Abaqus assume a hollow circular section. The internal stress caused by internal or external pressure loading in the pipe is included in these elements so that on the pipe cross-section a point under tension has different yield than a point under compression (*Figure 2*), thus causing an asymmetry in the section's response to inelastic bending. Two formulations are available for pipe elements in Abaqus. The thin-walled pipe formulation assumes constant hoop stress across the cross-section and neglects the radial stress, whereas thick-walled pipes (available only in Abaqus/Standard) allow the hoop and radial stress components to vary across the cross-section.

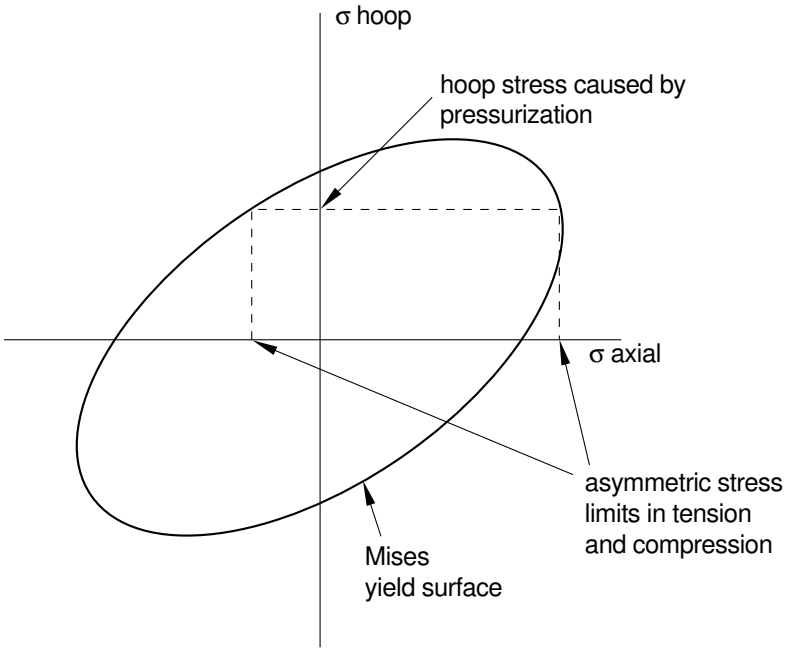


Figure 2: Yield behavior in thin-walled PIPE elements.

The hoop stress in thin-walled pipe elements is computed as the average stress in equilibrium with the internal and external pressure loading on the pipe section. For the thin-walled formulation, an integration rule with one point through the thickness suffices to obtain an accurate solution.

For thick-walled pipes, the hoop stress and radial stress variation under applied internal and/or external pressure are calculated using Lamé's equations. The constitutive calculations at each material point take into account the imposed hoop and radial stress values to determine the structural response. A two-dimensional integration rule is used for thick-walled pipes to capture the effect of stress variation across the section accurately.

"Hybrid" Beams

Hybrid beam element types (B21H, B33H, etc.) are provided in Abaqus/Standard for use in cases where it is numerically difficult to compute the axial and shear forces in the beam by the usual finite element displacement method. This problem arises most commonly in geometrically nonlinear analysis when the beam undergoes large rotations and is very rigid in axial and transverse shear deformation, such as a link in a vehicle's suspension system or a flexing long pipe or cable. The problem in such cases is that slight differences in nodal positions can cause very large forces, which, in turn, cause large motions in other directions. The hybrid elements overcome this difficulty by using a more general formulation in which the axial and transverse shear forces in the elements are included, along with the nodal displacements and rotations, as primary variables. Although this formulation makes these elements more expensive, they generally converge much faster when the beam's rotations are large and, therefore, are more efficient overall in such cases.

References

- Archer, J. S., "Consistent Matrix Formulations for Structural Analysis using Finite-Element Techniques," *American Institute of Aeronautics and Astronautics Journal*, vol. 3, pp. 1910–1918, 1965.
- Cowper, R. G., "The Shear Coefficient in Timoshenko's Beam Theory," *Journal of Applied Mechanics*, vol. 33, pp. 335–340, 1966.

Beam Element Cross-Section Orientation

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Beam Modeling*
- *Beam Cross-Section Library*
- *Beam Section Behavior*
- *Assigning a beam orientation*

Overview

The orientation of a beam cross-section:

- is defined in terms of a local, right-handed axis system; and
- can be user-defined or calculated by Abaqus.

Beam Cross-Sectional Axis System

The orientation of a beam cross-section is defined in Abaqus in terms of a local, right-handed ($\mathbf{t}$, $\mathbf{n_1}$, $\mathbf{n_2}$) axis system, where $\mathbf{t}$ is the tangent to the axis of the element, positive in the direction from the first to the second node of the element, and $\mathbf{n_1}$ and $\mathbf{n_2}$ are basis vectors that define the local 1- and 2-directions of the cross-section. $\mathbf{n_1}$ is referred to as the first beam section axis, and $\mathbf{n_2}$ is referred to as the normal to the beam. This beam cross-sectional axis system is illustrated in *Figure 1*.

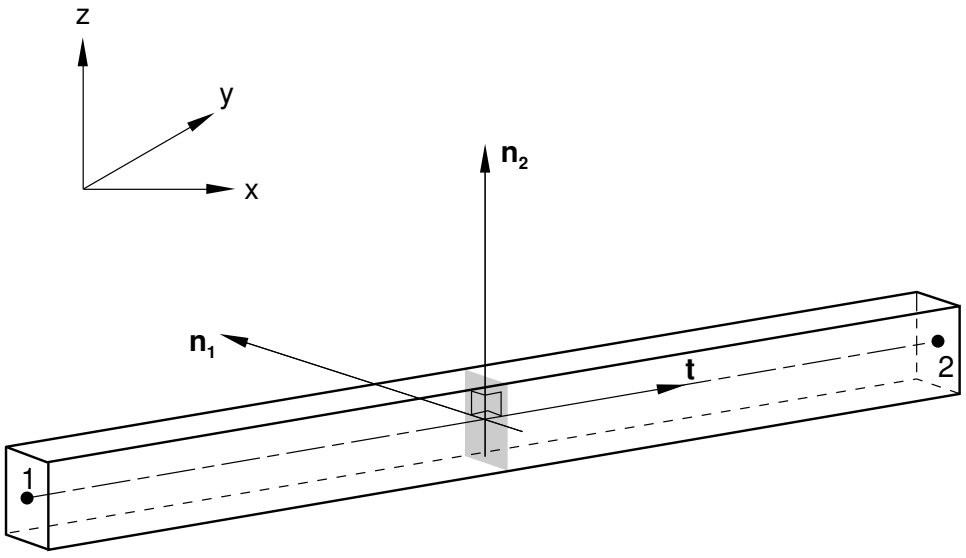


Figure 1: Local axis definition for beam-type elements.

Defining the N_1-Direction

For beams in a plane the $\mathbf{n_1}$-direction is always (0.0, 0.0, −1.0); that is, normal to the plane in which the motion occurs. Therefore, planar beams can bend only about the first beam-section axis.

For beams in space the approximate direction of $\mathbf{n_1}$ must be defined directly as part of the beam section definition or by specifying an additional node off the beam axis as part of the element definition (see *Element Definition*). This additional node is included in the element's connectivity list.

- If an additional node is specified, the approximate direction of $\mathbf{n_1}$ is defined by the vector extending from the first node of the element to the additional node.
- If $\mathbf{n_1}$ is defined directly for the section and an additional node is specified, the direction calculated by using the additional node will take precedence.
- If the approximate direction is not defined by either of the above methods, the default value is (0.0, 0.0, −1.0).

This approximate $\mathbf{n_1}$-direction can be used to determine the $\mathbf{n_2}$-direction (discussed below). Once the $\mathbf{n_2}$-direction has been defined or calculated, the actual $\mathbf{n_1}$-direction will be calculated as $\mathbf{n_2} \times \mathbf{t}$, possibly resulting in a direction that is different from the specified direction.

Input File Usage: Use the following option to specify the $\mathbf{n_1}$-direction directly for a beam section integrated during the analysis:

**BEAM SECTION*
$\mathbf{n_1}$*-direction (the data line number depends on the value of the SECTION parameter)*

Use the following option to specify the $\mathbf{n_1}$-direction directly for a general beam section:

**BEAM GENERAL SECTION*
$\mathbf{n_1}$*-direction (the data line number depends on the value of the SECTION parameter)*

Use the following option to specify an additional node off the beam axis to define the $\mathbf{n_1}$-direction:

**ELEMENT*

Abaqus/CAE Usage: Property module: **Assign->Beam Section Orientation**: select region and enter the $\mathbf{n_1}$-direction

Specifying an additional node off the beam axis is not supported in Abaqus/CAE.

Defining Nodal Normals

For beams in space you can define the nodal normal ($\mathbf{n_2}$-direction) by giving its direction cosines as the fourth, fifth, and sixth coordinates of each node definition or by giving them in a user-specified normal definition; see *Normal Definitions at Nodes* for details. Otherwise, the nodal normal will be calculated by Abaqus, as described below.

If the nodal normal is defined as part of the node definition, this normal is used for all of the structural elements attached to the node except those for which a user-specified normal is defined. If a user-specified normal is defined at a node for a particular element, this normal definition takes precedence over the normal defined as part of the node definition. If the specified normal subtends an angle that is greater than 20° with the plane perpendicular to the element axis, a warning message is issued in the data (`.dat`) file. If the angle between the normal defined as part of the node definition or the user-specified normal and $\mathbf{t} \times \mathbf{n_1}$ is greater than 90°, the reverse of the specified normal is used. This behavior corresponds to a "rule-based" reversal of user-specified

normals. If the default rule-based reversal of the nodal normal is unexpected, you can specify that the user-specified nodal normal remain fixed.

Input File Usage: Use the following option to specify the $\mathbf{n}_2$-direction as part of the node definition:

**NODE*
node number, nodal coordinates, nodal normal coordinates

Use the following option to define a user-specified normal (can be reversed):

**NORMAL*, BEAM NORMAL=RULE BASED

Use the following option to hold the user-specified normal fixed without any rule-based reversal:

**NORMAL*, BEAM NORMAL=FIXED

Abaqus/CAE Usage: Defining the nodal normal is not supported in Abaqus/CAE; the nodal normal calculated by Abaqus is always used.

Calculation of the Average Nodal Normals by Abaqus

If the nodal normal is not defined as part of the node definition, element normal directions at the node are calculated for all shell and beam elements for which a user-specified normal is not defined (the "remaining" elements). For shell elements the normal direction is orthogonal to the shell midsurface, as described in *About Shell Elements*. For beam elements the normal direction is the second cross-section direction, as described in *Beam Element Cross-Section Orientation*. The following algorithm is then used to obtain an average normal (or multiple averaged normals) for the remaining elements that need a normal defined:

1. If a node is connected to more than 30 remaining elements, no averaging occurs and each element is assigned its own normal at the node. The first nodal normal is stored as the normal defined as part of the node definition. Each subsequent normal is stored as a user-specified normal.
2. If a node is shared by 30 or fewer remaining elements, the normals for all the elements connected to the node are computed. Abaqus takes one of these elements and puts it in a set with all the other elements that have normals within 20° of it. Then:
   a. Each element whose normal is within 20° of the added elements is also added to this set (if it is not yet included).
   b. This process is repeated until the set contains for each element in the set all the other elements whose normals are within 20°.
   c. If all the normals in the final set are within 20° of each other, an average normal is computed for all the elements in the set. If any of the normals in the set are more than 20° out of line from even a single other normal in the set, no averaging occurs for elements in the set and a separate normal is stored for each element.
   d. This process is repeated until all the elements connected to the node have had normals computed for them.
   e. The first nodal normal is stored as the normal defined as part of the node definition. Each subsequently generated nodal normal is stored as a user-specified normal.

This algorithm ensures that the nodal averaging scheme has no element order dependence. A simple example illustrating this process is included below.

Example: Beam Normal Averaging

Consider the three beam element model in *Figure 2*. Elements 1, 2, and 3 share a common node 10, with no user-specified normal defined.

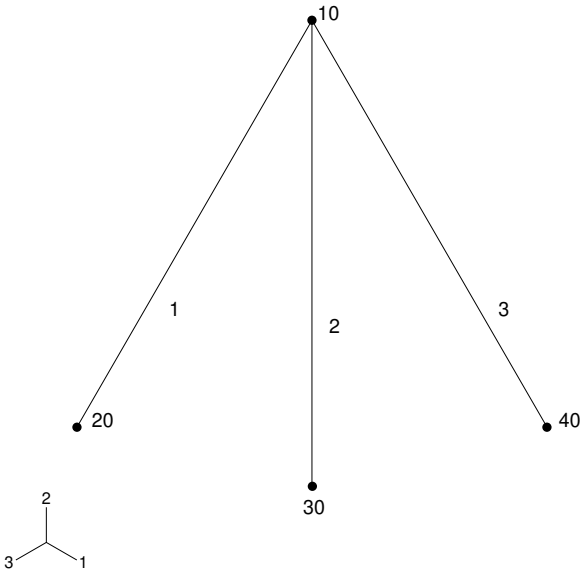


Figure 2: Three-element example for nodal averaging algorithm.

In the first scenario, suppose that at node 10 the normal for element 2 is within 20° of both elements 1 and 3, but the normals for elements 1 and 3 are not within 20° of each other. In this case, each element is assigned its own normal: one is stored as part of the node definition and two are stored as user-specified normals.

In the second scenario, suppose that at node 10 the normal for element 2 is within 20° of both elements 1 and 3 and the normals for elements 1 and 3 are within 20° of each other. In this case, a single average normal for elements 1, 2, and 3 would be computed and stored as part of the node definition.

In the last scenario, suppose that at node 10 the normal for element 2 is within 20° of element 1 but the normal of element 3 is not within 20° of either element 1 or 2. In this case, an average normal is computed and stored for elements 1, and 2 and the normal for element 3 is stored by itself: one is stored as part of the node definition and the other is stored as a user-specified normal.

Appropriate Beam Normals

To ensure proper application of loads that act normal to the beam cross-section, it is important to have beam normals that correctly define the plane of the cross-section. When linear beams are used to model a curved geometry, appropriate beam normals are the normals that are averaged at the nodes. For such cases it is preferable to define the cross-sectional axis system such that beam normals lie in the plane of curvature and are properly averaged at the nodes.

Initial Curvature and Initial Twist

In Abaqus/Standard normal direction definitions can result in a beam element having an initial curvature or an initial twist, which will affect the behavior of some elements.

- When the normal to an element is not perpendicular to the beam axis (obtained by interpolation using the nodes of the element), the beam element is curved. Initial curvature can result when you define the normal directly (as part of the node definition or as a user-specified normal) or can result when beams intersect at a node and the normals to the beams are averaged as described above. The effect of this initial curvature is considered in cubic beam elements. Initial curvature resulting from normal definitions is not considered in

quadratic beam elements; however, these elements do properly account for any initial curvature represented by the node positions.

- Similarly, nodal-normal directions that are in different orientations about the beam axis at different nodes imply a twist. The effect of an initial twist, which could result from normal averaging or user-defined normal definitions, is considered in quadratic beam elements.

Since the behavior of initially curved or initially twisted beams is quite different from straight beams, the changes caused by averaging the normals might result in changes in the deformation of some beam elements. You should always check the model to ensure that the changes caused by averaging the normals are intended. If the normal directions at successive nodes subtend an angle that is greater than 20°, a warning message is issued in the data (`.dat`) file. In addition, a warning message will be issued during input file preprocessing if the average curvature computed for a beam differs by more than 0.1 degrees per unit length or if the approximate integrated curvature for the entire beam differs by more than 5 degrees as compared to the curvature computed without nodal averaging and without user-defined normals.

In Abaqus/Explicit initial curvature of the beam is not taken into account: all beam elements are assumed to be initially straight. The element's cross-section orientation is calculated by averaging the $\mathbf{n_1}$- and $\mathbf{n_2}$-directions associated with its nodes. These two vectors are then projected onto the plane that is perpendicular to the beam element's axis. These projected directions $\mathbf{n_1}$ and $\mathbf{n_2}$ are made orthogonal to each other by rotating in this plane by an equal and opposite angle.

Beam Section Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Beam Modeling*
- **BEAM GENERAL SECTION*
- **BEAM SECTION*
- *Creating beam sections*

Overview

The beam section behavior:

- is defined in terms of the response of the beam section to stretching, bending, shear, and torsion;
- might or might not require numerical integration over the section; and
- can be linear or nonlinear (as a result of nonlinear material response).

Beam Section Behavior

Defining a beam section's response to stretching, bending, shear, and torsion of the beam's axis requires a suitable definition of the axial force, N; bending moments, M_{11} and M_{22}; and torque, T, as functions of the axial strain, ε; curvature changes, κ_{11} and κ_{22}; and twist, ϕ. Here the subscripts 1 and 2 refer to local, orthogonal axes in the beam section.

If open-section beam types are used, the section behavior must also define the warping *bimoment*, W, and the generalized strain measures include the warping amplitude, w, and the *bicurvature* of the beam, χ, which is the gradient of the warping amplitude with respect to position along the beam: $\chi = dw/dS$.

The type of section definition you choose determines whether the beam section properties are recomputed during the progression of the analysis or established in the preprocessor for the duration of the analysis. If a general beam section definition is used (see *Using a General Beam Section to Define the Section Behavior*), the cross-section properties are computed once, during preprocessing. Alternatively, a beam section definition that is integrated during the analysis can be used (see *Using a Beam Section Integrated during the Analysis to Define the Section Behavior*), in which case Abaqus will use numerical integration of the stress over the cross-section to define the beam's response as the analysis proceeds.

Since planar beams deform only in the X–Y plane, only N and M_{11}, and ε and κ_{11}, contribute to the response in these elements: κ_{22}, ϕ, and w are assumed to be zero.

In Abaqus bending moments in beam sections are typically measured about the centroid of the beam section, while torque is measured with respect to the shear center. In general, for meshed beam sections that exhibit full coupling between the strains, no unique centroid and shear center locations can be defined. Therefore, all section forces and section moments are given with respect to the beam axis for such cross-sections. The beam axis (defined as the line joining the nodes that define the beam element) need not pass through the centroid of the beam section.

The degrees of freedom of the beam element are at the origin of the local $(1, 2)$ coordinate system defined in the cross-section of the beam; that is, the line of the element connecting the element's nodes passes through the origin of the cross-section's local coordinate system.

Determining Whether to Use a Beam Section Integrated during the Analysis or a General Beam Section

When a beam section integrated during the analysis is used (see *Using a Beam Section Integrated during the Analysis to Define the Section Behavior*), Abaqus integrates numerically over the section as the beam deforms, evaluating the material behavior independently at each point on the section. This type of beam section should be used when the section nonlinearity is caused only by nonlinear material response.

When a general beam section is used (see *Using a General Beam Section to Define the Section Behavior*), Abaqus precomputes the beam cross-section quantities and performs all section computations during the analysis in terms of the precomputed values. This method combines the functions of beam section and material descriptions. The precomputed section properties might be specified in a variety of ways, including quite complex geometries defined with a two-dimensional finite element mesh (see *Meshed Beam Cross-Sections*). A general beam section should be used when the beam section response is linear or when it is nonlinear and the nonlinearity arises from more than just material nonlinearity, such as in cases when section collapse occurs.

Input File Usage: Use the following option to define a beam section integrated during the analysis:

**BEAM SECTION*

Use the following option to define a general beam section:

**BEAM GENERAL SECTION*

Abaqus/CAE Usage: To define a beam section integrated during the analysis:

Property module: **Create Section**: select **Beam** as the section **Category** and **Beam** as the section **Type**: **Section integration: During analysis**

To define a general beam section:

Property module: **Create Section**: select **Beam** as the section **Category** and **Beam** as the section **Type**: **Section integration: Before analysis**

Geometric Section Quantities

The section quantities described below are needed to define the behavior of a general beam section.

Moments of Inertia

The moments of inertia with respect to the centroid are defined as

$$I_{11} = \int_A (x_2 - x_2^c)^2 \, dA,$$

$$I_{22} = \int_A (x_1 - x_1^c)^2 \, dA, \text{ and}$$

$$I_{12} = \int_A (x_1 - x_1^c)\,(x_2 - x_2^c) \, dA,$$

where (x_1, x_2) is the position of the point in the local $(1, 2)$ beam section axis system and (x_1^c, x_2^c) is the position of the centroid of the cross-sectional area.

Bending stiffness and rotary inertia contributions for a meshed section profile (see *Meshed Cross-Sections*) are calculated using the two-dimensional cross-section model. The following integrated properties are defined for the entire cross-section model meshed with 1- DOF warping elements:

$$
\begin{aligned}
(EI)_{11} &= \int_A E(x_2 - x_2^c)^2 \, dA, \\
(EI)_{22} &= \int_A E(x_1 - x_1^c)^2 \, dA, \\
(EI)_{12} &= \int_A E\,(x_1 - x_1^c)\,(x_2 - x_2^c)\, dA, \\
(\rho I)_{11} &= \int_A \rho(x_2 - x_2^m)^2 \, dA, \\
(\rho I)_{22} &= \int_A \rho(x_1 - x_1^m)^2 \, dA, \text{ and} \\
(\rho I)_{12} &= \int_A \rho\,(x_1 - x_1^m)\,(x_2 - x_2^m)\, dA,
\end{aligned}
$$

where (x_1^m, x_2^m) is the center of mass of the cross section.

Torsional Constant

The torsional constant, *J*, depends on the shape of the cross section. The torsional constant of a circular section is the polar moment of inertia, $J = I_{11} + I_{22}$.

The torsional constant for the rectangular and trapezoidal library sections is calculated numerically by Abaqus using the Prandtl stress function approach. A local finite element model of the cross-section is created internally for this purpose. The number of integration points selected for the cross-section determines the accuracy of this finite element model. For increased accuracy specify a higher-order rule by selecting nondefault integration.

The above rule is also applied to both the thin-walled box section and the arbitrary section to increase the accuracy of the model. If the thickness for each segment making up the section varies significantly, more integration points for the box section or smaller segments for the arbitrary section should be specified in the area where the thickness varies.

The torsional stiffness for a meshed section is calculated over the two-dimensional region meshed with 1-DOF warping elements. The accuracy of the integration depends on the number of elements in the model:

$$(GJ) = \int_A G_{\alpha\beta}\left(\psi_{,\alpha} + \epsilon_\gamma^\alpha\left(x^\gamma - x_s^\gamma\right)\right)\left(\psi_{,\beta} + \epsilon_\delta^\beta\left(x^\delta - x_s^\delta\right)\right) dA,$$

where $\psi_{,\alpha}$ denotes the derivative of the warping function with respect to the cross-section (1, 2) axis and x_α^s is the position of the shear center of the cross-sectional area. All indices take values 1, 2. For more details, see *Meshed beam cross-sections*.

For closed thin-walled sections the torsional constant is calculated from

$$J = \frac{4A_c^2}{\oint 1/t \, ds},$$

where *t* is the thickness of the section, A_c is the area enclosed by the median line of the section, and *s* is the length of the median line, measured along the circumference of the section in a counterclockwise direction.

For open, built-up, thin-walled sections,

$$J = \int \frac{1}{3} t^3 \, ds.$$

Abaqus will check if a built-up section is closed or not and will use the appropriate torsional constant.

Sectorial Moment and Warping Constant

For open, thin-walled sections the sectorial moment is defined as

$$\Gamma_0 = \int_s S_w \, t \, ds$$

and the warping constant is defined as

$$\Gamma_W = \int_s S_w^2 \, t \, ds,$$

where S_w is the sectorial area at a point in the section with the shear center as its pole.

Rotary Inertia for Timoshenko Beams

In general, the rotary inertia associated with torsional modes is different from that of flexural modes. For unsymmetric cross-sections the rotary inertia is different in each direction of bending. For cross-sections where the beam node is not located at the center of mass, coupling exists between the translational and rotational degrees of freedom.

By default, the exact (anisotropic and coupled) rotary inertia is used for Timoshenko beams. In Abaqus/Standard the anisotropic rotary inertia introduces unsymmetric terms in the Jacobian operator during geometrically nonlinear, transient, direct-integration dynamic simulations. If the rotary inertia effects are significant in the geometrically nonlinear dynamic response and the exact rotary inertia is used, the unsymmetric solver should be used for better convergence.

Optionally, an approximate isotropic and uncoupled rotary inertia can be selected. In Abaqus/Standard this means that the rotary inertia associated with the torsional mode only is used for all rotational degrees of freedom; potentially destabilizing rotary inertia effects in impact problems due to the anisotropy or displacement-rotation coupling will not be introduced. In Abaqus/Explicit this means a scaled flexural inertia with a scaling factor chosen to maximize the stable element time increment is used for all rotational degrees of freedom; that is, the stable time increment is not determined by the flexural response of the beam. In some slender beam analyses an isotropic approximation to the rotary inertia might be accurate enough.

If beam elements are used to model plate-type structures in Abaqus/Explicit (that is, if the moment of inertia about one section axis of the beam is more than a thousand times greater than the moment of inertia about the other axis), the exact rotary inertia formulation might lead to a sharp cut-back in the stable time increment. In this case it is recommended that you either use the isotropic approximation or alternatively consider modeling the structure with shell elements, which might be better suited to this type of analysis.

For a definition of rotary inertia for the beam's cross-section, see *Mass and inertia for Timoshenko beams*.

Input File Usage: Use the following option to specify isotropic rotary inertia for a beam section integrated during the analysis:

**BEAM SECTION*, ROTARY INERTIA=ISOTROPIC

Use the following option to specify isotropic rotary inertia for a general beam section:

**BEAM GENERAL SECTION*, ROTARY INERTIA=ISOTROPIC

Abaqus/CAE Usage: Isotropic rotary inertia for beam sections is not supported in Abaqus/CAE. The default exact rotary inertia is always used.

Adding Inertia to the Beam Section Behavior for Timoshenko Beams

You can define additional mass and rotary inertia properties for Timoshenko beams (including PIPE elements) . This added inertia defined within the cross-section per unit length along the beam contributes to the inertia

response of the beam without contributing to the structural stiffness. However, you cannot define additional beam inertia for a section if you use isotropic rotary inertia or for beam elements with tapered cross-sections.

To specify additional beam inertia, you define the mass (per unit length) with the mass center positioned at point (x_1, x_2) in the local (1, 2) beam cross-section axis system. To include rotary inertia (per unit length), you can also define the angle α (in degrees) within the cross-section local (1, 2) system that positions the first axis of the rotary inertia coordinate system (X, Y) relative to the local 1-direction in the beam cross-section axis system. See *Figure 1* for an illustration.

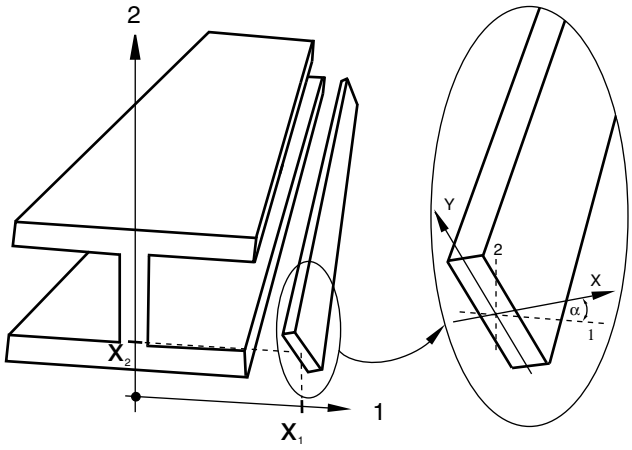


Figure 1: Beam element with added inertia.

The rotary inertia components relative to the rotary inertia coordinate system (X, Y) are defined as

$$I^{\rho}_{XX} = \int_A \rho Y^2 dA,$$
$$I^{\rho}_{YY} = \int_A \rho X^2 dA, \quad \text{and}$$
$$I^{\rho}_{XY} = -\int_A \rho XY dA,$$

where A is the area, ρ is the mass density, and X and Y are the local rotary inertia system coordinates measured from (x_1, x_2), the center of the added mass contribution.

As many point masses and rotary inertia contributions as are needed to define the added inertia can be specified. Mass proportional damping associated with the added inertia can be specified by assigning a value to the mass proportional Rayleigh damping coefficient, α_R, or the composite damping coefficient, ξ_α. Abaqus will use the mass weighted average (based on the material damping and the added inertia damping) for the element mass proportional damping.

Input File Usage: Use the following option in conjunction with the beam section definition to specify additional inertia properties:

**BEAM ADDED INERTIA*, ALPHA=α_R, COMPOSITE=ξ_α
mass per unit length, x_1, x_2, α, I_{11}, I_{22}, I_{12}

Abaqus/CAE Usage: Additional inertia properties are not supported in Abaqus/CAE.

Additional Inertia due to Immersion in Fluid

When a beam is fully or partially submerged, the effect of the surrounding fluid can be modeled as an additional distributed inertia on the beam (see *Loading due to an incident dilatational wave field*). By default, the beam is assumed to be fully submerged. Alternatively, you can specify that the added inertia per unit length should be reduced by a factor of one-half to model a partially submerged beam.

You specify the fluid mass density, ρ_f (per unit volume); beam local x and y coordinates of the wetted cross-section centroid; wetted section effective radius, r; and empirical drag or flow coefficients, C_A and C_{A-E}. The inertia added per unit length to a fully immersed beam cross-section is given by

$$\pi r^2 \rho_f C_A.$$

Because the beam cross-section origin might not be coincident with the centroid of the wetted cross-section, the additional fluid inertia might include rotary effects. Nonzero values for the x- and y-offsets of the wetted cross-section centroid will produce rotation-displacement coupling in the inertia formulation. The default model for the added inertia derives from inviscid flow around a cylindrical cross-section ($C_A = 1.0$); you can specify a coefficient, C_A, that models flow around a different cross-section geometry.

An immersed beam also experiences an additional added mass effect at its free ends. If a beam element's end node is not attached to any other element and additional fluid inertia is defined for this element, an additional mass might be added in the form:

$$\frac{8}{3} r^3 \rho_f C_{A-E}.$$

For $C_{A-E} = 1.0$ this added mass corresponds to that of a hemispherical cap; the default value is $C_{A-E} = 0.0$.

The coefficient C_{A-E} can be changed to model other geometries. If the beam is partially submerged, the end inertia is automatically reduced by one-half. However, the added mass at the free ends is always isotropic: axial and transverse motions experience the same additional inertia.

The “virtual mass” added to a submerged or partially submerged beam is not included in the total mass, center of mass, moments, or products of inertia reported in the data (`.dat`) file.

Input File Usage: Use the following option in conjunction with the beam section definition to define a fully immersed beam:

**BEAM FLUID INERTIA*, FULL

ρ_f, x, y, r, C_A, C_{A-E}

Use the following option in conjunction with the beam section definition to define a partially immersed beam:

**BEAM FLUID INERTIA*, HALF

ρ_f, x, y, r, C_A, C_{A-E}

Abaqus/CAE Usage: To define a fully immersed beam:

Property module: beam section editor: **Fluid Inertia**: toggle on **Specify fluid inertia effects**: **Fully submerged**

To define a partially immersed beam:

Property module: beam section editor: **Fluid Inertia**: toggle on **Specify fluid inertia effects**: **Half submerged**

References

- Blevins, R. D., Formulas for Natural Frequency and Mode Shape, R. E. Krieger Publishing Co., Inc. , 1987.

Using a Beam Section Integrated during the Analysis to Define the Section Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Beam Modeling*
- *Beam Section Behavior*
- **BEAM SECTION*
- **BEAM SECTION OFFSET*
- *Specifying properties for beam sections integrated during analysis*

Overview

A beam section integrated during the analysis:

- is used when section properties must be recomputed as the beam deforms over the course of the analysis; and
- can be associated with linear or nonlinear material behavior.

Defining the Behavior of a Beam Section Integrated during the Analysis

Use a beam section integrated during the analysis to define the section behavior when numerical integration over the section is required as the beam deforms. You can choose a section shape from the library of beam section shapes provided (see *Beam Cross-Section Library*) and define the section's dimensions. In addition, you can specify the number of section points to use for integration. The default number of section points is adequate for monotonic loading that causes plasticity. If reversed plasticity occurs, more section points are required.

You can also specify an offset for the section origin, as illustrated in *Figure 1*, for a rectangular cross-section. In *Figure 1*, the original cross-section axes $\mathbf{n}_1$-$\mathbf{n}_2$ are always placed at the beam axis, while the local 1-2 system defines the shifted cross-section axes after offsets of x_{1o} and x_{2o} are applied in the $\mathbf{n}_1$- and $\mathbf{n}_2$-directions, respectively.

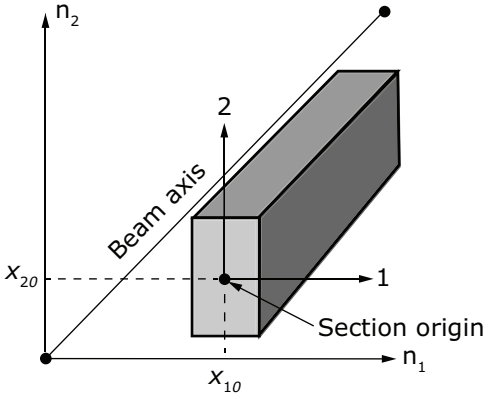


Figure 1: Offset for the section origin.

Use a material definition (*Material Data Definition*) to define the material properties of the section, and associate these properties with the section definition. Linear or nonlinear material behavior can be associated with the section definition. However, if the material response is linear, the more economic approach is to use a general beam section (see *Using a General Beam Section to Define the Section Behavior*).

You must associate the section properties with a region of your model.

Input File Usage: **BEAM SECTION*, ELSET=*name*, SECTION=*library_section*, MATERIAL=*name*

The ELSET parameter is used to associate the section properties with a set of beam elements.

Include the following option in conjunction with the **BEAM SECTION* option to specify an offset for the section origin:

**BEAM SECTION OFFSET*

Abaqus/CAE Usage: Property module:
Create Profile: **Name:** *library_section*
Create Section: select **Beam** as the section **Category** and **Beam** as the section **Type**, **Section integration: During analysis**, **Profile name:** *library_section*, **Material name:** *name*, Offset for origin of cross-section: specify X1 and X2 values
Assign->Section: select regions

Defining a Change in Cross-Sectional Area due to Straining

In the shear flexible elements Abaqus provides for a possible uniform cross-sectional area change by allowing you to specify an effective Poisson's ratio for the section. This effect is considered only in geometrically nonlinear analysis (see *Defining an Analysis*) and is provided to model the reduction or increase in the cross-sectional area for a beam subjected to large axial stretch.

The value of the effective Poisson's ratio must be between −1.0 and 0.5. By default, this effective Poisson's ratio for the section is set to 0.0 so that this effect is ignored. Setting the effective Poisson's ratio to 0.5 implies that the overall response of the section is incompressible. This behavior is appropriate if the beam is made of a typical metal whose overall response at large deformation is essentially incompressible (because it is dominated by plasticity). Values between 0.0 and 0.5 mean that the cross-sectional area changes proportionally between no change and incompressibility, respectively. A negative value of the effective Poisson's ratio results in an increase in the cross-sectional area in response to tensile axial strains.

This effective Poisson's ratio is not available for use with Euler-Bernoulli beam elements.

Input File Usage: **BEAM SECTION*, POISSON=ν_{eff}

Abaqus/CAE Usage: Property module: **Create Section**: select **Beam** as the section **Category** and **Beam** as the section **Type**: **Section integration: During analysis**, **Section Poisson's ratio:** ν_{eff}

Defining Material Damping

When a beam section integrated during the analysis is used, damping can be introduced through the material behavior definition. See *Material Damping* for more information about the material damping types available in Abaqus.

Specifying Temperature and Field Variables

Temperature and field variables can be specified at specific points through the section or by defining the value at the origin of the cross-section and specifying the gradients in the local 1- and 2-directions. The actual values of the temperature and field variables are specified as either predefined fields or initial conditions (see *Predefined Fields* or *Initial Conditions*).

In any element it is assumed that the temperature definitions at all the nodes of the element are compatible with the temperature definition method chosen for the element. For cases in which the temperature definition method changes from one element to the next, separate nodes must be used on the interface between elements with different temperature definition methods and MPCs must be applied to make the displacements and rotations the same at the nodes.

By Defining the Value at the Origin and the Gradients in the 1- and 2-Directions

Temperatures and field variables can be defined by giving the value at the origin of the cross-section and the gradients in the 2- and 1-directions of the cross-section (that is, give $\theta, \partial\theta/\partial X_2,$ and $\partial\theta/\partial X_1$ in the predefined field or initial condition definition). For beams in a plane only θ and $\partial\theta/\partial X_2$ need be given; gradients in the 1-direction are ignored in this case.

Input File Usage: **BEAM SECTION*, TEMPERATURE=GRADIENTS

Abaqus/CAE Usage: Property module: **Create Section**: select **Beam** as the section **Category** and **Beam** as the section **Type**: **Section integration: During analysis**, **Linear by gradients**

By Defining the Values at Points through the Section

Temperatures and field variables can be defined at a set of points on the section, as indicated for each cross-section in *Beam Cross-Section Library*.

This technique cannot be used for any beam element that is adjacent to a general beam section element, as it can lead to incorrect temperature distributions at the shared cross-section. If you cannot avoid this modeling scenario, you must define the adjacent elements using separate nodes connected by MPCs, as discussed above.

Input File Usage: **BEAM SECTION*, TEMPERATURE=VALUES

Abaqus/CAE Usage: Property module: **Create Section**: select **Beam** as the section **Category** and **Beam** as the section **Type**: **Section integration: During analysis**, **Interpolated from temperature points**

Output

Beam section properties such as cross-sectional area, moments of inertia, etc. are printed in the model data output. When a beam section integrated during the analysis is used, section forces, moments, and transverse shear forces and section strains, curvatures, and transverse shear strains can be output for the section (see *Element Output* and *Writing Element Output to the Output Database*). In addition, stress and strain can be output at each section point. *Beam Element Library* lists some of the element output quantities that are available for beam elements.

Axial strains due to warping are included in the stress/strain output from Abaqus/Standard if a beam section integrated during the analysis is used.

Temperature output at the section points can be obtained using the element variable TEMP. If the temperatures are given at specific points through the section, output at the temperature points can be obtained using the nodal variable NT*xx*. The nodal variable NT*xx* should not be used for output at the temperature points if the temperatures are specified by defining the value at the origin of the cross-section and specifying the gradients in the local 1- and 2-directions. In this case output variable NT should be requested; NT11 (the reference temperature value) and NT12 and NT13 (the temperature gradients in the local 1- and 2-directions, respectively) are output automatically.

Beam normals are written to the output database automatically for all frames that include field output of nodal displacements. The normal directions can be visualized in the Visualization module of Abaqus/CAE.

Using a General Beam Section to Define the Section Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Beam Modeling*
- *Beam Section Behavior*
- **BEAM GENERAL SECTION*
- **BEAM SECTION OFFSET*
- *Specifying properties for general beam sections*

Overview

A general beam section:

- is used to define beam section properties that are computed once and held constant for the entire analysis;
- can be used to define linear or nonlinear section behavior;
- for linear section behavior can be associated only with linear material behavior;
- enables the use of meshed cross-sections (*Meshed Beam Cross-Sections*); and
- enables the use of tapered cross-sections (Abaqus/Standard only).

Linear Section Behavior

Linear section response is calculated as follows. At each point in the cross-section the axial stress, σ, and the shear stress, τ, are given by

$$\sigma = E\left(\overline{\theta}, f_\beta\right)\ \left(\varepsilon - \varepsilon^{th}\right)\ \text{and}\ \tau = G\left(\overline{\theta}, f_\beta\right)\gamma,$$

where

$E\left(\overline{\theta}, f_\beta\right)$

is Young's modulus (which may depend on the temperature, $\overline{\theta}$, and field variables, f_β, at the beam axis);

$G\left(\overline{\theta}, f_\beta\right)$

is the shear modulus (which may also depend on the temperature and field variables at the beam axis);

ε

is the axial strain;

γ

is the shear caused by twist; and

ε^{th}

is the thermal expansion strain.

The thermal expansion strain is given by

$$\varepsilon^{th} = \alpha\left(\overline{\theta}, f_\beta\right)\ \left(\theta - \theta^0\right) - \alpha\left(\overline{\theta}^I, f_\beta^I\right)\ \left(\theta^I - \theta^0\right),$$

where

$\alpha\left(\overline{\theta}, f_\beta\right)$

is the thermal expansion coefficient,

θ

is the current temperature at a point in the beam section,

f_β

are field variables,

θ^0

is the reference temperature for α,

θ^I

is the initial temperature at this point (see *Defining Initial Temperatures*), and

f_β^I

are the initial values of the field variables at this point (see *Defining Initial Values of Predefined Field Variables*).

If the thermal expansion coefficient is temperature or field-variable dependent, it is evaluated at the temperature and field variables at the beam axis. Therefore, since we assume that θ varies linearly over the section, ε^{th} also varies linearly over the section.

The temperature is defined from the temperature of the beam axis and the gradients of temperature with respect to the local x_1- and x_2-axes:

$$\theta = \overline{\theta} + \frac{\partial \theta}{\partial x_1} x_1 + \frac{\partial \theta}{\partial x_2} x_2.$$

The axial force, *N*; bending moments, M_1 and M_2 about the 1 and 2 beam section local axes; torque, *T*; and bimoment, *W*, are defined in terms of the axial stress σ and the shear stress τ (see *Beam element formulation*). These terms are

$$
\begin{aligned}
N &= E\left(A\left(\varepsilon_c - \varepsilon_c^{th}\right) + \Gamma_0\chi\right),\\
M_1 &= E\left(I_{11}\left(\kappa_1 - \frac{\partial\varepsilon^{th}}{\partial x_2}\right) - I_{12}\left(\kappa_2 + \frac{\partial\varepsilon^{th}}{\partial x_1}\right)\right),\\
M_2 &= E\left(-I_{12}\left(\kappa_1 - \frac{\partial\varepsilon^{th}}{\partial x_2}\right) + I_{22}\left(\kappa_2 + \frac{\partial\varepsilon^{th}}{\partial x_1}\right)\right),\\
T &= GJ\phi + GI_p w_p,\\
W &= E\left(\Gamma_0\left(\varepsilon_c - \varepsilon_c^{th}\right) + \Gamma_W\chi\right),
\end{aligned}
$$

where

A

is the area of the section,

I_{11}

is the moment of inertia for bending about the 1-axis of the section,

I_{12}

is the moment of inertia for cross-bending,

I_{22}

is the moment of inertia for bending about the 2-axis of the section,

J

is the torsional constant,

Γ_0

is the sectorial moment of the section,

Γ_W

is the warping constant of the section,

ε_c

is the axial strain measured at the centroid of the section,

ε_c^{th}

is the thermal axial strain,

κ_1

is the curvature change about the first beam section local axis,

κ_2

is the curvature change about the second beam section local axis,

ϕ

is the twist,

χ

is the bicurvature defining the axial strain in the section due to the twist of the beam, and

$w_p = w_f - w$

is the difference between the unconstrained warping amplitude, w_f, and the actual warping amplitude, w.

Γ_0, Γ_W, χ, and w_p are nonzero only for open-section beam elements.

Defining Linear Section Behavior for Library Cross-Sections or Linear Generalized Cross-Sections

Linear beam section response is defined geometrically by A, I_{11}, I_{12}, I_{22}, J, and (if necessary) Γ_0 and Γ_W.

You can input these geometric quantities directly or specify a standard library section and Abaqus calculates these quantities. In either case define the orientation of the beam section (see *Beam Element Cross-Section Orientation*).

You can specify Young's modulus, the shear modulus, and the coefficient of thermal expansion as functions of temperature; and associate the section properties with a region of your model. If the thermal expansion coefficient is temperature dependent, the reference temperature for thermal expansion must also be defined as described later in this section.

Alternatively, you can associate a material definition (*Material Data Definition*) with the section definition. Abaqus determines the equivalent section properties. You must associate the section behavior with a region of your model.

Specifying the Geometric Quantities Directly

You can define “generalized” linear section behavior by specifying A, I_{11}, I_{12}, I_{22}, J, and—if necessary—Γ_0 and Γ_W directly. In this case you can specify the location of the centroid, thus allowing the bending axis of the beam to be offset from the line of its nodes. In addition, you can specify the location of the shear center.

Input File Usage: Use the following option to define generalized linear beam section properties:

**BEAM GENERAL SECTION*, SECTION=GENERAL, ELSET=*name*
A, I_{11}, I_{12}, I_{22}, J, Γ_0, Γ_W

If necessary, use the following option to specify the location of the centroid:

**CENTROID*

If necessary, use the following option to specify the location of the shear center:

**SHEAR CENTER*

Abaqus/CAE Usage: Property module:
Create Profile: **Name:** *generalized_section*, **Generalized**
Create Section: select **Beam** as the section **Category** and **Beam** as the section **Type**: **Section integration: Before analysis**, **Profile name:** *generalized_section*: **Centroid** and **Shear Center**
Assign->Section: select regions

Specifying a Standard Library Section and Allowing Abaqus to Calculate the Geometric Quantities

You can select one of the standard library sections (see *Beam Cross-Section Library*) and specify the geometric input data required to define the shape of the cross-section. Abaqus then calculates the geometric quantities required to define the section behavior automatically.

In addition, you can specify an offset for the section origin, as illustrated in *Figure 1*, for a rectangular cross-section. In *Figure 1*, the original cross-section axes $\mathbf{n}_1$-$\mathbf{n}_2$ are always placed at the beam axis, while the local 1-2 system defines the shifted cross-section axes after offsets of x_{1o} and x_{2o} are applied in the $\mathbf{n}_1$- and $\mathbf{n}_2$-directions, respectively.

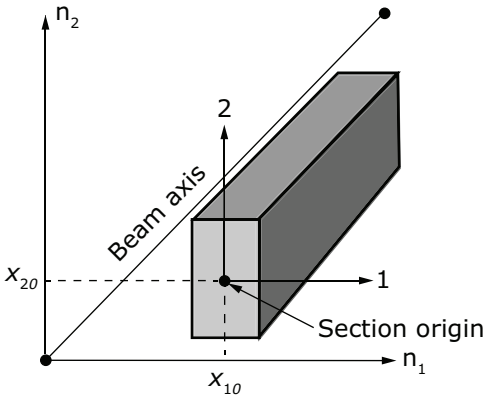


Figure 1: Offset for the section origin.

Input File Usage: **BEAM GENERAL SECTION*, SECTION=*library_section*, ELSET=*name*

Include the following option in conjunction with the **BEAM SECTION* option to specify an offset for the section origin:

**BEAM SECTION OFFSET*

Abaqus/CAE Usage: Property module:
Create Profile: **Name:** *library_section*
Create Section: select **Beam** as the section **Category** and **Beam** as the section **Type**, **Section integration: Before analysis**, **Profile name:** *library_section*, Offset for origin of cross-section: specify X1 and X2 values
Assign->Section: select regions

Specifying the Linear Section Response with a Material Definition

The material definition (*Material Data Definition*) can contain isotropic linear elastic behavior (*Linear Elastic Behavior*) and isotropic thermal expansion behavior (*Thermal Expansion*). If both the isotropic linear elastic material behavior and the isotropic thermal expansion behavior are temperature or field variable dependent, the

values of the independent variables (temperature or field variables) that you specify must be the same for both the linear elastic moduli and thermal expansion coefficient. You can specify damping behavior (*Material Damping*). In Abaqus/Explicit you must define the density (*Density*) of the material. In an Abaqus/Standard analysis the density is needed only when the mass of the beam elements is required. Any nonlinear material properties (such as plastic behavior) are ignored.

Input File Usage: **BEAM GENERAL SECTION*, MATERIAL=*name*, ELSET=*name*

Abaqus/CAE Usage: Property module:
Create Profile: **Name:** *library_section*
Create Section: select **Beam** as the section **Category** and **Beam** as the section **Type**: **Section integration: Before analysis**; **Profile name:** *library_section*; **Material definition type: Material name**: *material_name*
Assign->Section: select regions

Defining Linear Section Behavior for Meshed Cross-Sections

Linear beam section response for a meshed section profile is obtained by numerical integration from the two-dimensional model. The numerical integration is performed once, determining the beam stiffness and inertia quantities, as well as the coordinates of the centroid and shear center, for the duration of the analysis. These beam section properties are calculated during the beam section generation and are written to the text file *jobname*`.bsp`. This text file can be included in the beam model. See *Meshed Beam Cross-Sections* for a detailed description of the properties defining the linear beam section response for a meshed section, as well as for how a typical meshed section is analyzed.

Input File Usage: Use the following options:

**BEAM GENERAL SECTION*, SECTION=MESHED, ELSET=*name*
**INCLUDE*, INPUT=*jobname*`.bsp`

Abaqus/CAE Usage: Meshed cross-sections are not supported in Abaqus/CAE.

Defining Linear Section Behavior for Tapered Cross-Sections in Abaqus/Standard

In Abaqus/Standard you can define Timoshenko beams with linearly tapered cross-sections. General beam sections with linear response and standard library sections are supported, with the exception of arbitrary sections. The section parameters are defined at the two end nodes of each beam element. The effective beam area and moment of inertia for bending about the 1- and 2-axis of the section used in the calculation of the beam stiffness matrix, section forces, and stresses are

$$A^{\text{eff}} = \frac{A^I + \sqrt{A^I A^J} + A^J}{3},$$

$$I_{11}^{\text{eff}} = \frac{I_{11}^I + \sqrt[4]{\left(I_{11}^I\right)^3 I_{11}^J} + \sqrt{I_{11}^I I_{11}^J} + \sqrt[4]{I_{11}^I \left(I_{11}^J\right)^3} + I_{11}^J}{5},$$

$$I_{22}^{\text{eff}} = \frac{I_{22}^I + \sqrt[4]{\left(I_{22}^I\right)^3 I_{22}^J} + \sqrt{I_{22}^I I_{22}^J} + \sqrt[4]{I_{22}^I \left(I_{22}^J\right)^3} + I_{22}^J}{5},$$

where the superscripts I and J refer to the two end nodes of the beam. The remaining effective geometric quantities are calculated as the average between the values at the two end nodes. This approximation suffices

for mild tapering along each element, but it can lead to large errors if the tapering is not gradual. Abaqus/Standard issues a warning message during input file preprocessing if the area or inertia ratio is larger than 2.0 and an error message if the ratio is larger than 10.0.

The effective area and inertia are not used in the computation of the mass matrix. Instead, terms on the diagonal quadrants use the properties from the respective nodes, while off-diagonal quadrants use averaged quantities.

For example, the axial inertia a linear element would have the diagonal term coming from node I of $\rho A^I/3$, while node J contributes with $\rho A^J/3$ and the two off-diagonal contributions equal $\rho\left(A^I+A^J\right)/12$. Mild tapering is assumed in this formulation, since the total mass of the element totals $\rho\left(A^I+A^J\right)/2$.



Note: When you apply a tapered beam section to geometry in Abaqus/CAE, the full tapering is applied to each element along the beam's length. For beams that include multiple elements, this modeling style can create a "sawtooth" pattern along the length of the beam. If you want to model gradual tapering along the entire length of the beam in Abaqus/CAE, you must calculate the size and shape of the beam profiles at the intermediate nodes, then apply different tapered beam sections to each beam element along the length.

Input File Usage: Use the following option to define linear section behavior of tapered cross-sections:

**BEAM GENERAL SECTION*, TAPER, ELSET=*name*

Abaqus/CAE Usage: Property module:
Create Profile: **Name:** *library_section*
Create Section: select **Beam** as the section **Category** and **Beam** as the section **Type**: **Section integration: Before analysis**, **Beam shape along length**: **Tapered**: **Beam start** and **Beam end** options: **Profile name:** *library_section*
Assign->Section: select regions

Nonlinear Section Behavior

Typically nonlinear section behavior is used to include the experimentally measured nonlinear response of a beam-like component whose section distorts in its plane. When the section behaves according to beam theory (that is, the section does not distort in its plane) but the material has nonlinear response, it is usually better to use a beam section integrated during the analysis to define the section geometrically (see *Using a Beam Section Integrated during the Analysis to Define the Section Behavior*), in association with a material definition.

Nonlinear section behavior can also be used to model beam section collapse in an approximate sense: *Nonlinear dynamic analysis of a structure with local inelastic collapse* illustrates this for the case of a pipe section that may suffer inelastic collapse due to the application of a large bending moment. In following this approach you should recognize that such unstable section collapse, like any unstable behavior, typically involves localization of the deformation: results, therefore, are strongly mesh sensitive.

Calculation of Nonlinear Section Response

Nonlinear section response is assumed to be defined by

$$\begin{aligned} N &= N\left(\varepsilon_c-\varepsilon_c^{th},\bar{\theta},f_\beta\right),\\ M_1 &= M_1\left(\kappa_1,\bar{\theta},f_\beta\right),\\ M_2 &= M_2\left(\kappa_2,\bar{\theta},f_\beta\right),\\ T &= T\left(\phi,\bar{\theta},f_\beta\right), \end{aligned}$$

where $(\quad)$ means a functional dependence on the conjugate variables: $N = N(\varepsilon)$, $M_1 = M_1(\kappa_1)$, etc. For example, $N\left(\varepsilon_c - \varepsilon_c^{th}, \bar{\theta}, f_\beta\right)$ means that N is a function of:$\varepsilon_c - \varepsilon_c^{th}$; $\bar{\theta}$, the temperature of the beam axis; and of f_β, any predefined field variables at the beam axis. When the section behavior is defined in this way, only the temperature and field variables of the beam axis are used: any temperature or field-variable gradients given across the beam section are ignored.

These nonlinear responses may be purely elastic (that is, fully reversible—the loading and unloading responses are the same, even though the behavior is nonlinear) or may be elastic-plastic and, therefore, irreversible.

The assumption that these nonlinear responses are uncoupled is restrictive; in general, there is some interaction between these four behaviors, and the responses are coupled. You must determine if this approximation is reasonable for a particular case. The approach works well if the response is dominated by one behavior, such as bending about one axis. However, it may introduce additional errors if the response involves combined loadings.

Defining Nonlinear Section Behavior

You can define "generalized" nonlinear section behavior by specifying the area, A; moments of inertia, I_{11} for bending about the 1-axis of the section, I_{22} for bending about the 2-axis of the section, and I_{12} for cross-bending; and torsional constant, J. These values are used only to calculate the transverse shear stiffness; and, if needed, A is used to compute the mass density of the element. In addition, you can define the orientation and the axial, bending, and torsional behavior of the beam section (N, M_1, M_2, T), as well as the thermal expansion coefficient. If the thermal expansion coefficient is temperature dependent, the reference temperature for thermal expansion must also be defined as described below.

Nonlinear generalized beam section behavior cannot be used with beam elements with warping degrees of freedom.

The axial, bending, and torsional behavior of the beam section and the thermal expansion coefficient are defined by tables. See *Material Data Definition* for a detailed discussion of the tabular input conventions. In particular, you must ensure that the range of values given for the variables is sufficient for the application since Abaqus assumes a constant value of the dependent variable outside this range.

Input File Usage: Use the following options to define generalized nonlinear beam section properties:

**BEAM GENERAL SECTION*, SECTION=NONLINEAR GENERAL, ELSET=*name*

A, I_{11}, I_{12}, I_{22}, J
**AXIAL* for N

**M1* for M_1

**M2* for M_2
**TORQUE* for T
**THERMAL EXPANSION* for the thermal expansion coefficient

Abaqus/CAE Usage: Nonlinear generalized cross-sections are not supported in Abaqus/CAE.

Defining Linear Response for N, M_1, M_2, and T

If the particular behavior is linear, N, M_1, M_2, and T should be specified as functions of the temperature and predefined field variables, if appropriate.

As an example of axial behavior, if

$$N = (AE)\ \left(\varepsilon_c - \varepsilon_c^{th}\right),$$

where (AE) is constant for a given temperature, the value of (AE) is entered. (AE) can still be varied as a function of temperature and field variables.

Input File Usage: Use the following options to define linear axial, bending, and torsional behavior:

**AXIAL*, LINEAR
**M1*, LINEAR
**M2*, LINEAR
**TORQUE*, LINEAR

Abaqus/CAE Usage: Nonlinear generalized cross-sections are not supported in Abaqus/CAE.

Defining Nonlinear Elastic Response for N, M_1, M_2, and T

If the particular behavior is nonlinear but elastic, the data should be given from the most negative value of the kinematic variable to the most positive value, always giving a point at the origin. See *Figure 2* for an example.

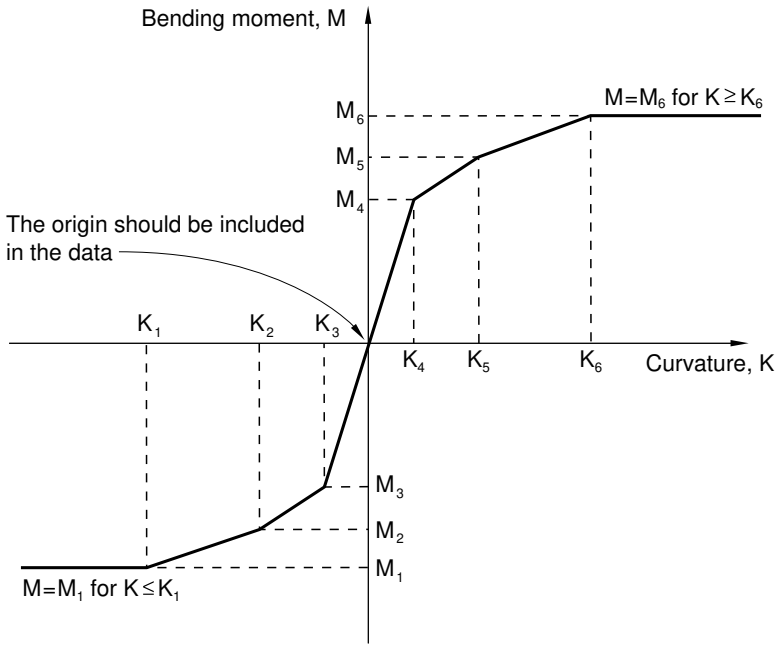


Figure 2: Example of elastic nonlinear beam section behavior definition.

Input File Usage: Use the following options to define nonlinear elastic axial, bending, and torsional behavior:

**AXIAL*, ELASTIC
**M1*, ELASTIC
**M2*, ELASTIC
**TORQUE*, ELASTIC

Abaqus/CAE Usage: Nonlinear generalized cross-sections are not supported in Abaqus/CAE.

Defining Elastic-Plastic Response for N, M_1, M_2, and T

By default, elastic-plastic response is assumed for N, M_1, M_2, and T.

The inelastic model is based on assuming linear elasticity and isotropic hardening (or softening) plasticity. The data in this case must begin with the $(0, 0)$ point and proceed to give positive values of the kinematic variable at increasing positive values of the conjugate force or moment. Strain softening is allowed. The elastic modulus is defined by the slope of the initial line segment, so that straining beyond the point that terminates that initial line segment is partially inelastic. If strain reversal occurs in that part of the response, it is elastic initially. See *Figure 3* for an example.

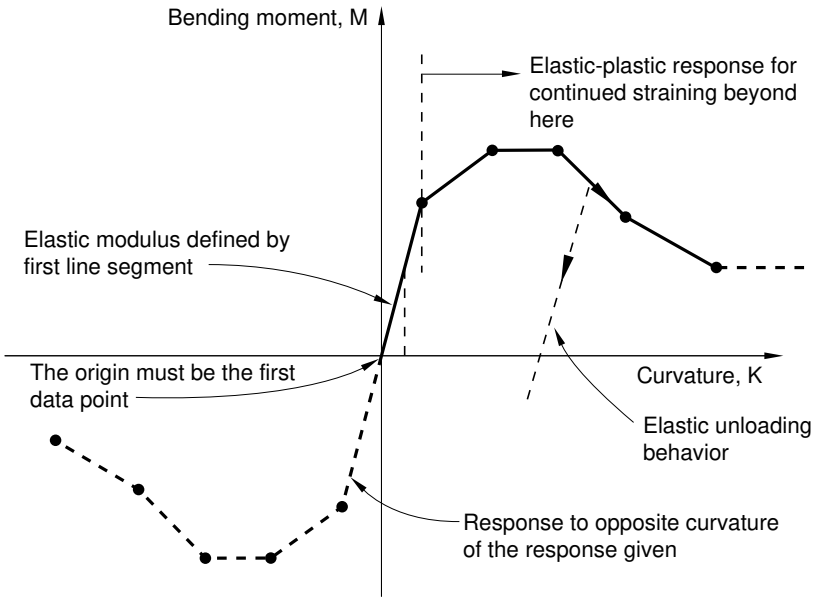


Figure 3: Example of inelastic nonlinear beam section behavior definition.

Input File Usage: Use the following options to define elastic-plastic axial, bending, and torsional behavior:

**AXIAL*
**M1*
**M2*
**TORQUE*

Abaqus/CAE Usage: Nonlinear generalized cross-sections are not supported in Abaqus/CAE.

Defining the Reference Temperature for Thermal Expansion

The thermal expansion coefficient may be temperature dependent. In this case the reference temperature for thermal expansion, θ^0, must be defined.

Input File Usage: **BEAM GENERAL SECTION*, ZERO=θ^0

Abaqus/CAE Usage: Property module: **Create Section**: select **Beam** as the section **Category** and **Beam** as the section **Type**: **Section integration: Before analysis**: **Basic**: **Specify reference temperature:** θ^0

Defining the Initial Section Forces and Moments

You can define initial stresses (see *Defining Initial Stresses*) for general beam sections that are applied as initial section forces and moments. Initial conditions can be specified only for the axial force, the bending moments, and the twisting moment. Initial conditions cannot be prescribed for the transverse shear forces.

Defining a Change in Cross-Sectional Area due to Straining

In the shear flexible elements Abaqus provides for a possible uniform cross-sectional area change by allowing you to specify an effective Poisson's ratio for the section. This effect is considered only in geometrically nonlinear analysis (see *Defining an Analysis*) and is provided to model the reduction or increase in the cross-sectional area for a beam subjected to large axial stretch.

The value of the effective Poisson's ratio must be between −1.0 and 0.5. By default, this effective Poisson's ratio for the section is set to 0.0 so that this effect is ignored. Setting the effective Poisson's ratio to 0.5 implies that the overall response of the section is incompressible. This behavior is appropriate if the beam is made of rubber or if it is made of a typical metal whose overall response at large deformation is essentially incompressible (because it is dominated by plasticity). Values between 0.0 and 0.5 mean that the cross-sectional area changes proportionally between no change and incompressibility, respectively. A negative value of the effective Poisson's ratio results in an increase in the cross-sectional area in response to tensile axial strains.

This effective Poisson's ratio is not available for use with Euler-Bernoulli beam elements.

Input File Usage: **BEAM GENERAL SECTION*, POISSON=ν_{eff}

Abaqus/CAE Usage: Property module: **Create Section**: select **Beam** as the section **Category** and **Beam** as the section **Type**: **Section integration: Before analysis**: **Basic**: **Section Poisson's ratio:** ν_{eff}

Defining Damping

When the beam section and material behavior are defined by a general beam section, you can include mass and viscous stiffness proportional damping in the dynamic response (calculated in Abaqus/Standard with the direct time integration procedure, *Implicit Dynamic Analysis Using Direct Integration*).

See *Material Damping* for more information about the material damping types available in Abaqus.

Input File Usage: Use both of the following options:

**BEAM GENERAL SECTION*
**DAMPING*

Abaqus/CAE Usage: Property module: **Create Section**: select **Beam** as the section **Category** and **Beam** as the section **Type**: **Section integration: Before analysis**: **Damping**: **Alpha**, **Beta**, **Structural**, and **Composite**

Specifying Temperature and Field Variables

Define temperatures and field variables by giving the values at the origin of the cross-section as either predefined fields or initial conditions (see *Predefined Fields* or *Initial Conditions*). Temperature gradients can be specified in the local 1- and 2-directions; other field-variable gradients defined through the cross-section are ignored in the response of beam elements that use a general beam section definition.

Output

Only the section forces, moments, and transverse shear forces and section strains, curvatures, and transverse shear strains can be output (see *Element Output* and *Writing Element Output to the Output Database*).

You can output stress and strain at particular points in the section. For linear section behavior defined using a standard library section or a generalized section, only axial stress and axial strain values are available. For linear section behavior defined using a meshed section, axial and shear stress and strain are available. For nonlinear generalized section behavior, axial strain output only is provided.

Specifying the Output Section Points for Standard Library Sections and Generalized Sections

To locate points in the section at which output of axial strain (and, for linear section behavior, axial stress) is

required, specify the local (x_1, x_2) coordinates of the point in the cross-section with respect to the section origin in the local coordinate system (that is, the 1-2 axes in *Figure 1*) of the cross-section: Abaqus numbers the points 1, 2, … in the order that they are given.

The variation of ε over the section is given by

$$\varepsilon = \varepsilon_c + \kappa_1 (x_2 - x_{2c}) - \kappa_2 (x_1 - x_{1c}),$$

where (x_{1c}, x_{2c}) are the local coordinates of the centroid of the beam section and κ_1 and κ_2 are the changes of curvature for the section.

For open-section beam element types, the variation of ε over the section has an additional term of the form

$\psi(x_1, x_2)\chi$, where $\psi(x_1, x_2)$ is the warping function. The warping function itself is undefined in the general beam section definition. Therefore, Abaqus will not take into account the axial strain due to warping when calculating section points output. Axial strains due to warping are included in the stress/strain output if a beam section integrated during the analysis is used.

Abaqus uses St. Venant torsion theory for noncircular solid sections. The torsion function and its derivatives are necessary to calculate shear stresses in the plane of the cross-section. The function and its derivatives are not stored for a general beam section. Therefore, you can request output of axial components of stress/strain only. A beam section integrated during the analysis must be used to obtain output of shear stresses.

Input File Usage: Use both of the following options to specify the output section points for general beam sections:

**BEAM GENERAL SECTION*
**SECTION POINTS*
x_1, x_2, ...

Abaqus/CAE Usage: Property module: **Create Section**: select **Beam** as the section **Category** and **Beam** as the section **Type**: **Section integration: Before analysis**: **Output Points**: **x1**, **x2**, ...

Requesting Output of Maximum Axial Stress/Strain in Abaqus/Standard

If you specify the output section points to obtain the maximum axial stress/strain (MAXSS) for a linear generalized section, the output value is the maximum of the values at the user-specified section points. You must select enough section points to ensure that this is the true maximum. MAXSS output is not available for nonlinear generalized sections or for an Abaqus/Explicit analysis.

Specifying the Output Section Points for Meshed Cross-Sections

For meshed cross-sections you can indicate in the two-dimensional cross-section analysis the elements and integration points where the stress and strain will be calculated during the subsequent beam analysis. Abaqus will then add the section points specification to the resulting *jobname*`.bsp` text file. This text file is then included as the data for the general beam section definition in the subsequent beam analysis. See *Meshed Beam Cross-Sections* for details.

The variation of the axial strain ε over the meshed section is given by

$$\varepsilon = \varepsilon_c + \kappa_1 \left(x_2 - x_{2c}\right) - \kappa_2 \left(x_1 - x_{1c}\right),$$

where $\left(x_{1c}, x_{2c}\right)$ are the local coordinates of the centroid of the beam section and κ_1 and κ_2 are the changes of curvature for the section.

The variations of shear components γ_1 and γ_2 over the meshed section are given by

$$\gamma_1 = \gamma_{s1} + \phi\left(\frac{\partial \Psi}{\partial x_1} - \left(x_2 - x_{2s}\right)\right),$$
$$\gamma_2 = \gamma_{s2} + \phi\left(\frac{\partial \Psi}{\partial x_2} + \left(x_1 - x_{1s}\right)\right),$$

where $\left(x_{1s}, x_{2s}\right)$ are the local coordinates of the shear center of the beam section, ϕ is the twist of the beam axis, $\Psi\left(x_1, x_2\right)$ is the warping function, and γ_{s1} and γ_{s2} are shear strains due to the transverse shear forces.

For the case of an orthotropic composite beam material, the axial stress σ and the two shear components τ_1 and τ_2 are calculated in the beam section (1, 2) axis as follows:

$$\begin{Bmatrix} \sigma \\ \tau_1 \\ \tau_2 \end{Bmatrix} = \begin{bmatrix} E & 0 & 0 \\ & G_1(\cos\alpha)^2 + G_2(\sin\alpha)^2 & (G_1 - G_2)\cos\alpha\sin\alpha \\ & sym & G_1(\sin\alpha)^2 + G_2(\cos\alpha)^2 \end{bmatrix} \begin{Bmatrix} \varepsilon \\ \gamma_1 \\ \gamma_2 \end{Bmatrix},$$

where α determines the material orientation.

Input File Usage: Use both of the following options in the two-dimensional meshed cross-section analysis to specify the output section points for the subsequent beam analysis:

**BEAM SECTION GENERATE*
**SECTION POINTS*
section_point_label, *element_number*, *integration_point_number*

Abaqus/CAE Usage: Meshed cross-sections are not supported in Abaqus/CAE.

Beam Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the beam elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *About Beam Modeling*
- *Choosing a Beam Element*
- **BEAM GENERAL SECTION*
- **BEAM SECTION*

Element Types

Beams in a Plane

B21	2-node linear beam
B21H$^{(S)}$	2-node linear beam, hybrid formulation
B22	3-node quadratic beam
B22H$^{(S)}$	3-node quadratic beam, hybrid formulation
B23$^{(S)}$	2-node cubic beam
B23H$^{(S)}$	2-node cubic beam, hybrid formulation
PIPE21	2-node linear pipe
PIPE21H$^{(S)}$	2-node linear pipe, hybrid formulation
PIPE22$^{(S)}$	3-node quadratic pipe
PIPE22H$^{(S)}$	3-node quadratic pipe, hybrid formulation

Active Degrees of Freedom

1, 2, 6

Additional Solution Variables

All of the cubic beam elements have two additional variables relating to axial strain.

The linear thin-walled pipe elements have one additional variable, and the quadratic thin-walled pipe elements have two additional variables relating to the hoop strain. The linear thick-walled pipe elements have two additional variables, and the quadratic thick-walled pipe elements have four additional variables relating to the hoop and radial strain components.

The hybrid beam and pipe elements have additional variables relating to the axial force and transverse shear force. The linear elements have two, the quadratic elements have four, and the cubic elements have three additional variables.

Beams in Space

B31	2-node linear beam
B31H$^{(S)}$	2-node linear beam, hybrid formulation
B32	3-node quadratic beam
B32H$^{(S)}$	3-node quadratic beam, hybrid formulation
B33$^{(S)}$	2-node cubic beam
B33H$^{(S)}$	2-node cubic beam, hybrid formulation
PIPE31	2-node linear pipe
PIPE31H$^{(S)}$	2-node linear pipe, hybrid formulation
PIPE32$^{(S)}$	3-node quadratic pipe
PIPE32H$^{(S)}$	3-node quadratic pipe, hybrid formulation

Active Degrees of Freedom

1, 2, 3, 4, 5, 6

Additional Solution Variables

All of the cubic beam elements have two additional variables relating to axial strain.

The linear thin-walled pipe elements have one additional variable, and the quadratic thin-walled pipe elements have two additional variables relating to the hoop strain. The linear thick-walled pipe elements have two additional variables, and the quadratic thick-walled pipe elements have four additional variables relating to the hoop and radial strain components.

The hybrid beam and pipe elements have additional variables relating to the axial force and transverse shear force in the linear and quadratic elements and to the axial force only in the cubic elements. The linear and cubic elements have three and the quadratic elements have six additional variables.

Open-Section Beams in Space

B31OS$^{(S)}$	2-node linear beam
B31OSH$^{(S)}$	2-node linear beam, hybrid formulation
B32OS$^{(S)}$	3-node quadratic beam
B32OSH$^{(S)}$	3-node quadratic beam, hybrid formulation

Active Degrees of Freedom

1, 2, 3, 4, 5, 6, 7

Additional Solution Variables

Element type B31OSH has three additional variables and element type B32OSH has six additional variables relating to the axial force and transverse shear force.

Nodal Coordinates Required

Beams in a plane: X, Y, also (optional) N_x, N_y, the direction cosines of the normal.

Beams in space: X, Y, Z, also (optional) N_x, N_y, N_z, the direction cosines of the second local cross-section axis.

Element Property Definition

For PIPE elements use the pipe section type to specify the thin-walled pipe formulation or the thick pipe section type to specify the thick-walled pipe formulation. No other section types can be used with PIPE elements.

For open-section elements use only the arbitrary, channel, hat, I, L, and linear generalized section types.

Local orientations defined as described in *Orientations* cannot be used with beam elements to define local material directions. The orientation of the local beam section axes in space is discussed in *Beam Element Cross-Section Orientation*.

Input File Usage: Use either of the following options:

**BEAM SECTION*
**BEAM GENERAL SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Beam** as the section **Category** and **Beam** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): CENT$^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2} ($ML^{-1}T^{-2}$)

Description: Centrifugal force (magnitude is input as $m\omega^2$, where m is the mass per unit length and ω is the angular velocity).

Load ID (*DLOAD): CENTRIF$^{(S)}$
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): CORIO(S)
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-2}T$ ($ML^{-1}T^{-1}$)

Description: Coriolis force (magnitude is input as $m\omega$, where m is the mass per unit length and ω is the angular velocity). The load stiffness due to Coriolis loading is not accounted for in direct steady-state dynamics analysis.

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}

Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): PX
Abaqus/CAE Load/Interaction: Line load
Units: FL^{-1}

Description: Force per unit length in global *X*-direction.

Load ID (*DLOAD): PY
Abaqus/CAE Load/Interaction: Line load
Units: FL^{-1}

Description: Force per unit length in global *Y*-direction.

Load ID (*DLOAD): PZ
Abaqus/CAE Load/Interaction: Line load
Units: FL^{-1}

Description: Force per unit length in global *Z*-direction (only for beams in space).

Load ID (*DLOAD): PXNU
Abaqus/CAE Load/Interaction: Line load
Units: FL^{-1}

Description: Nonuniform force per unit length in global *X*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): PYNU
Abaqus/CAE Load/Interaction: Line load
Units: FL^{-1}

Description: Nonuniform force per unit length in global *Y*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): PZNU
Abaqus/CAE Load/Interaction: Line load
Units: FL^{-1}

Description: Nonuniform force per unit length in global *Z*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. (Only for beams in space.)

Load ID (*DLOAD): P1
Abaqus/CAE Load/Interaction: Line load
Units: FL^{-1}

Description: Force per unit length in beam local 1-direction (only for beams in space).

Load ID (*DLOAD): P2

Abaqus/CAE Load/Interaction: Line load
Units: FL^{-1}

Description: Force per unit length in beam local 2-direction.

Load ID (*DLOAD): P1NU
Abaqus/CAE Load/Interaction: Line load
Units: FL^{-1}

Description: Nonuniform force per unit length in beam local 1-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. (Only for beams in space.)

Load ID (*DLOAD): P2NU
Abaqus/CAE Load/Interaction: Line load
Units: FL^{-1}

Description: Nonuniform force per unit length in beam local 2-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): ROTA$^{(S)}$
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Rotary acceleration load (magnitude is input as $\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is the rotary acceleration).

Load ID (*DLOAD): ROTDYNF$^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: T^{-1}

Description: Rotordynamic load (magnitude is input as $\boldsymbol{\omega}$, where $\boldsymbol{\omega}$ is the angular velocity).

The following load types are available only for PIPE elements:

Load ID (*DLOAD): HPI
Abaqus/CAE Load/Interaction: Pipe pressure
Units: FL^{-2}

Description: Hydrostatic internal pressure (closed-end condition), varying linearly with the global *Z*-coordinate.

Load ID (*DLOAD): HPE
Abaqus/CAE Load/Interaction: Pipe pressure
Units: FL^{-2}

Description: Hydrostatic external pressure (closed-end condition), varying linearly with the global *Z*-coordinate.

Load ID (*DLOAD): PI
Abaqus/CAE Load/Interaction: Pipe pressure
Units: FL^{-2}

Description: Uniform internal pressure (closed-end condition).

Load ID (*DLOAD): PE
Abaqus/CAE Load/Interaction: Pipe pressure
Units: FL^{-2}

Description: Uniform external pressure (closed-end condition).

Load ID (*DLOAD): PENU
Abaqus/CAE Load/Interaction: Pipe pressure
Units: FL^{-2}

Description: Nonuniform external pressure (closed-end condition) with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): PINU
Abaqus/CAE Load/Interaction: Pipe pressure
Units: FL^{-2}
Description: Nonuniform internal pressure (closed-end condition) with magnitude supplied via user subroutine *DLOAD*.

Abaqus/Aqua Loads

Abaqus/Aqua loads are specified as described in *Abaqus/Aqua Analysis*. They are not available for open-section beams and do not apply to beams that are defined to have additional inertia due to immersion in fluid (see *Additional Inertia due to Immersion in Fluid*). In Abaqus/Explicit, Aqua loads can be applied only on linear beam and pipe elements.

Load ID (*CLOAD/ *DLOAD): FDD
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Transverse fluid drag load.

Load ID (*CLOAD/ *DLOAD): FD1
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Fluid drag force on the first end of the beam (node 1).

Load ID (*CLOAD/ *DLOAD): FD2
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Fluid drag force on the second end of the beam (node 2 or node 3).

Load ID (*CLOAD/ *DLOAD): FDT
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Tangential fluid drag load.

Load ID (*CLOAD/ *DLOAD): FI
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Transverse fluid inertia load.

Load ID (*CLOAD/ *DLOAD): FI1
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Fluid inertia force on the first end of the beam (node 1).

Load ID (*CLOAD/ *DLOAD): FI2
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Fluid inertia force on the second end of the beam (node 2 or node 3).

Load ID (*CLOAD/ *DLOAD): PB
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}

Description: Buoyancy load (closed-end condition).

Load ID (*CLOAD/ *DLOAD): WDD
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Transverse wind drag load.

Load ID (*CLOAD/ *DLOAD): WD1
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Wind drag force on the first end of the beam (node 1).

Load ID (*CLOAD/ *DLOAD): WD2
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Wind drag force on the second end of the beam (node 2 or node 3).

Foundations

Foundations are available only in Abaqus/Standard and are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): $FX^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Stiffness per unit length in global *X*-direction.

Load ID (*FOUNDATION): $FY^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Stiffness per unit length in global *Y*-direction.

Load ID (*FOUNDATION): $FZ^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Stiffness per unit length in global *Z*-direction (only for beams in space).

Load ID (*FOUNDATION): $F1^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Stiffness per unit length in beam local *1*-direction (only for beams in space).

Load ID (*FOUNDATION): $F2^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Stiffness per unit length in beam local *2*-direction.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): P

Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-1}

Description: Force per unit length in beam local 2-direction. The distributed surface force is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-1}

Description: Nonuniform force per unit length in beam local 2-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The distributed surface force is positive in the direction opposite to the surface normal.

Incident Wave Loading

Incident wave loading is also available for these elements, with some restrictions. See *Acoustic and Shock Loads*.

Element Output

See *Beam Cross-Section Library* for a description of the beam element output locations.

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors, except for meshed sections, have the same components. For example, the stress components are as follows:

S11 Axial stress.

S22 Hoop stress (available only for pipe elements).

S33 Radial stress (available only for thick-walled pipe elements).

S12 Shear stress caused by torsion (available only for beam-type elements in space). This component is not available when thin-walled, open sections are employed (I-section, L-section, and arbitrary open section).

Stress and strain for section points for meshed sections

S11 Axial stress.

S12 Shear stress along the second cross-section axis caused by shear force and, for beam elements in space, torsion.

S13 Shear stress along the first cross-section axis caused by shear force and torsion (available only for beams in space).

Section Forces, Moments, and Transverse Shear Forces

SF1 Axial force.

SF2 Transverse shear force in the local 2-direction (not available for B23, B23H, B33, B33H).

SF3 Transverse shear force in the local 1-direction (available only for beams in space, not available for B33, B33H).

SM1 Bending moment about the local 1-axis.

SM2 Bending moment about the local 2-axis (available only for beams in space).

SM3 Twisting moment about the beam axis (available only for beams in space).

BIMOM Bimoment due to warping (available only for open-section beams in space).

ESF1 Effective axial force for beams subjected to pressure loading (available for all Abaqus/Standard stress/displacement analysis types except response spectrum and random response).

See *Beam element formulation* for the definitions of the section forces and moments.

The effective axial section force for beams subjected to pressure loading is defined as

$$\mathrm{ESF1} = \mathrm{SF1} + p_e A_e - p_i A_i,$$

where p_e and p_i are the external and the internal pressures, respectively, and A_e and A_i are the external and the internal pipe areas as defined in the load definition. The pressure loadings (with a closed-end condition) that are relevant to the effective axial force are external/internal pressure (load types PE, PI, PENU, and PINU); external/internal hydrostatic pressure (load types HPE and HPI); and, in an Abaqus/Aqua environment, buoyancy pressure, PB, which includes dynamic pressure if waves are present.

For beams that are not subjected to pressure loading, the effective axial force ESF1 is equal to the usual axial force SF1.

Section Strains, Curvatures, and Transverse Shear Strains

SE1 Axial strain.

SE2 Transverse shear strain in the local 2-direction (not available for B23, B23H, B33, and B33H).

SE3 Transverse shear strain in the local 1-direction (available only for beams in space, not available for B33 and B33H).

SK1 Curvature change about the local 1-axis.

SK2 Curvature change about the local 2-axis (available only for beams in space).

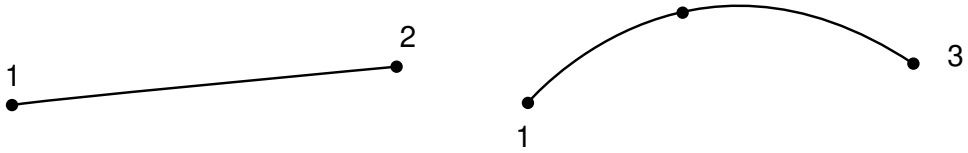
SK3 Twist of the beam (available only for beams in space).

BICURV Bicurvature due to warping (available only for open-section beams in space).

Beam Radius

BRADIUS Original beam radius for a solid circular beam.

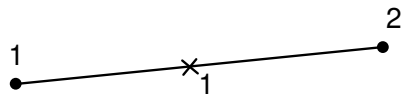
Node Ordering on Elements



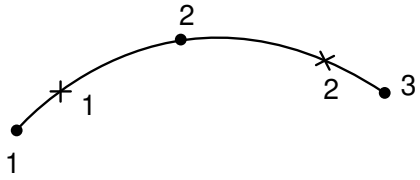
2 - node element 3 - node element

For beams in space an additional node may be given after a beam element's connectivity (in the element definition—see *Element Definition*) to define the approximate direction of the first cross-section axis, $\mathbf{n_1}$. See *Beam Element Cross-Section Orientation* for details.

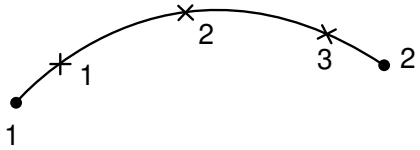
Numbering of Integration Points for Output



2 - node element



3 - node quadratic element



2 - node cubic element

Beam Cross-Section Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

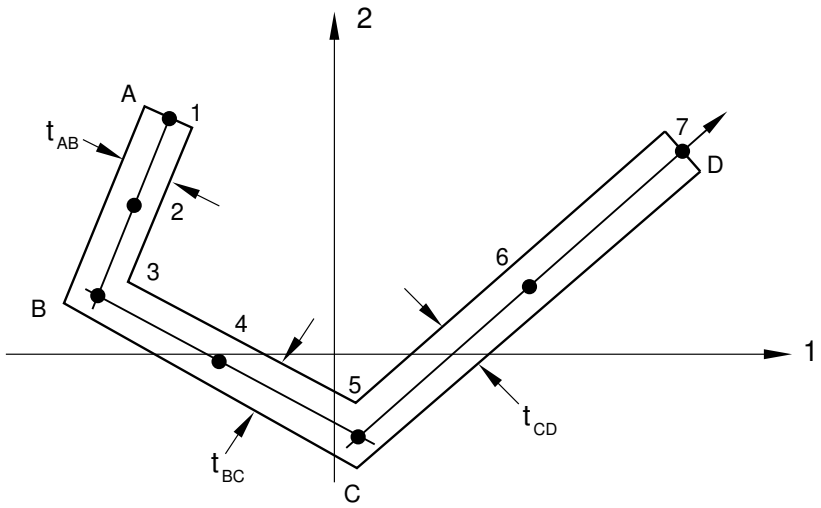
Overview

This section describes the standard beam sections that are available in Abaqus/Standard and Abaqus/Explicit for use with beam elements. A subset of the standard beam sections is available for use with frame elements in Abaqus/Standard. General (nonstandard) beam cross-sections can be defined as described in *Choosing a Beam Cross-Section*.

References:

- *About Beam Modeling*
- *Choosing a Beam Cross-Section*
- *Frame Elements*
- *Defining profiles*

Arbitrary, Thin-Walled, Open and Closed Sections



Example of arbitrary section

The arbitrary section type is provided to permit modeling of simple, arbitrary, thin-walled, open and closed sections. You specify the section by defining a series of points in the thin-walled cross-section of the beam; these points are then linked by straight line segments, each of which is integrated numerically along the axis of the section so that the section can be used together with nonlinear material behavior. An independent thickness is associated with each of the segments making up the arbitrary section.

Warping effects are included when an arbitrary section is used with open-section beam elements (available only in Abaqus/Standard).

Input File Usage: Use either of the following options:

**BEAM SECTION*, SECTION=ARBITRARY
**BEAM GENERAL SECTION*, SECTION=ARBITRARY

Abaqus/CAE Usage: Property module: **Create Profile**: **Arbitrary**

Restrictions

- An arbitrary section can be used only with beams in space (three-dimensional models).
- An arbitrary section should not be used to define closed sections with branches, multiply connected closed sections, or open sections with disconnected regions.
- For each individual segment of an arbitrary section there is no bending stiffness about the line joining the end points of the segment. Thus, an arbitrary section cannot be made up of only one segment.

Geometric Input Data

First, give the number of segments, the local coordinates of points *A* and *B*, and the thickness of the segment connecting these two vertices. Then, proceed by giving the local coordinates of point *C* and the thickness of the segment between points *B* and *C*, followed by the local coordinates of point *D* and the thickness of the segment between points *C* and *D*, and so on. An arbitrary section can contain as many segments as needed. All coordinates of section definition points are given in the local 1–2 axis system of the section.

The origin of the local 1–2 axis system is the beam node, and the position of this node used to define the section is arbitrary: it does not have to be the centroid.

Defining a Closed Section

A closed section is defined by making the starting and end points coincident. Only single-cell closed sections can be modeled accurately. Closed sections with fins (single branches attached to the cell) cannot be modeled with the capability in Abaqus.

Defining an Arbitrary Section with Discontinuous Branches

If the arbitrary section contains discontinuous sections (branches), a section with zero thickness should be used to return from the ending point of the branch to the starting point of the subsequent section. This zero thickness section should always coincide with a nonzero thickness section. For an example of an I-section defined using this method, see *Buckling analysis of beams*.

Default Integration

A three-point Simpson integration scheme is used for each segment making up the section. For more detailed integration, specify several segments along each straight portion of the section.

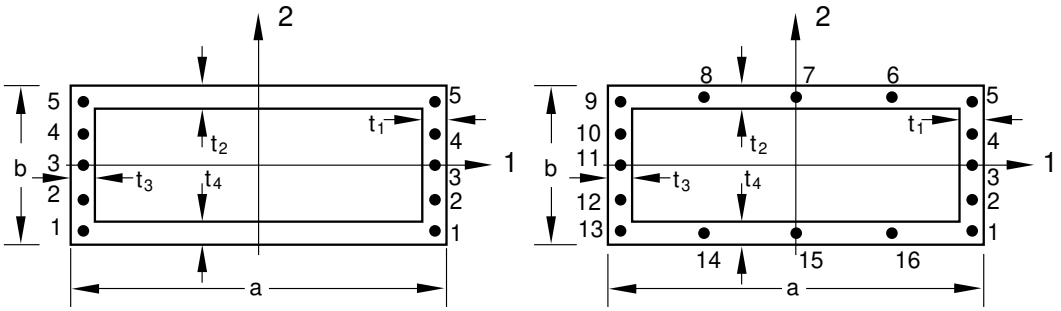
Default Stress Output Points If a Beam Section Integrated during the Analysis Is Used

The vertices of the section.

Temperature and Field Variable Input at Specific Points through Beam Sections Integrated during the Analysis

Give the value at each vertex of the section (points *A*, *B*, *C*, *D* in the figure).

Box Section



Default integration, beam in a plane

Default integration, beam in space

Input File Usage: Use one of the following options:

**BEAM SECTION*, SECTION=BOX
**BEAM GENERAL SECTION*, SECTION=BOX
**FRAME SECTION*, SECTION=BOX

Abaqus/CAE Usage: Property module: **Create Profile**: **Box**

Geometric Input Data

a, b, t_1, t_2, t_3, t_4

Default Integration (Simpson)

Beam in a plane: 5 points

Beam in space: 5 points in each wall (16 total)

Nondefault Integration Input for a Beam Section Integrated during the Analysis

Beam in a plane: Give the number of points in each wall that is parallel to the 2-axis. This number must be odd and greater than or equal to three.

Beam in space: Give the number of points in each wall that is parallel to the 2-axis, then the number of points in each wall that is parallel to the 1-axis. Both numbers must be odd and greater than or equal to three.

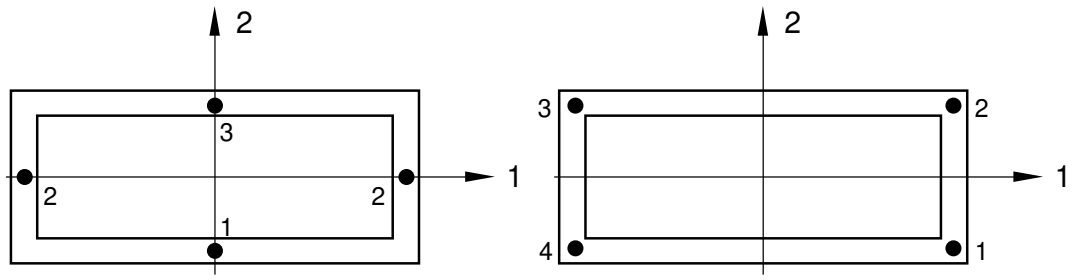
Default Stress Output Points If a Beam Section Integrated during the Analysis Is Used

Beam in a plane: Bottom and top (points 1 and 5 above for default integration).

Beam in space: 4 corners (points 1, 5, 9, and 13 above for default integration).

Temperature and Field Variable Input at Specific Points for Beam Sections Integrated during the Analysis

Give the value at each of the points shown below.

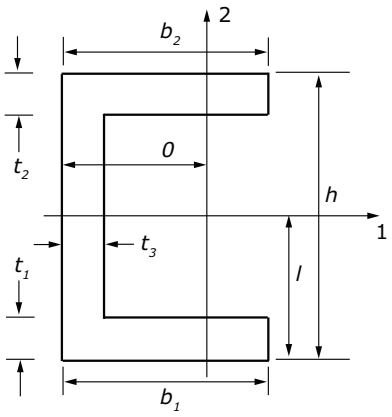


Beam in a plane Beam in space

Temperature Input for a Frame Section

Constant temperature throughout the element cross-section is assumed; therefore, only one temperature value per node is required.

Channel Section



Input File Usage: Use the following option:

**BEAM GENERAL SECTION*, SECTION=CHANNEL

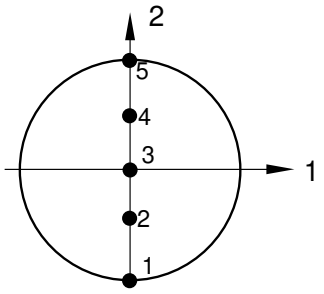
Abaqus/CAE Usage: Defining a channel section is not supported in Abaqus/CAE.

Geometric Input Data

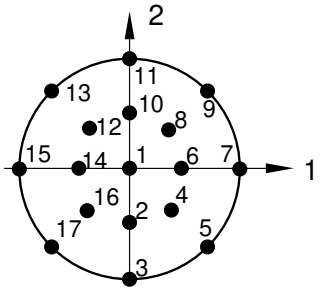
l, h, $\boldsymbol{b_1}$, $\boldsymbol{b_2}$, $\boldsymbol{t_1}$, $\boldsymbol{t_2}$, $\boldsymbol{t_3}$, o

You can specify l and o to place the origin of the local cross-section anywhere on the local 1- and local 2-axes. In the figure above, a negative value of l implies that the origin of the local cross-section is below the lower edge of the bottom flange, which might be required when constraining a beam stiffener to a shell. A negative value of o implies that the origin of the local cross-section is on the left of the leftmost outer edge of a channel web, which might be required when applying loads through the section shear center.

Circular Section



Default integration, beam in a plane



Default integration, beam in space

Input File Usage: Use one of the following options:

**BEAM SECTION*, SECTION=CIRC
**BEAM GENERAL SECTION*, SECTION=CIRC
**FRAME SECTION*, SECTION=CIRC

Abaqus/CAE Usage: Property module: **Create Profile**: **Circular**

Geometric Input Data

Radius

Default Integration

Beam in a plane: 5 points

Beam in space: 3 points radially, 8 circumferentially (17 total; trapezoidal rule). Integration point 1 is situated at the center of the beam and is used for output purposes only. It makes no contribution to the stiffness of the element; therefore, the integration point volume (IVOL) associated with this point is zero.

Nondefault Integration Input for a Beam Section Integrated during the Analysis

Beam in a plane: A maximum of 9 points are permitted.

Beam in space: Give an odd number of points in the radial direction, then an even number of points in the circumferential direction.

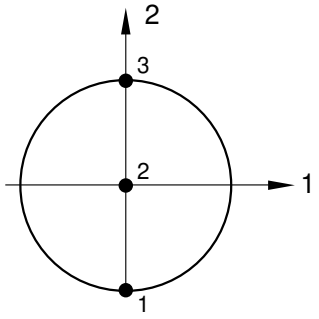
Default Stress Output Points If a Beam Section Integrated during the Analysis Is Used

Beam in a plane: Bottom and top (points 1 and 5 above for default integration).

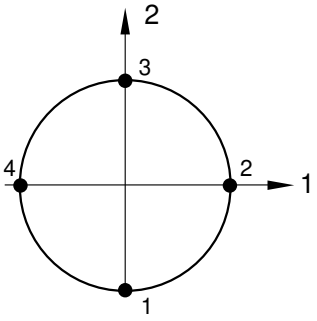
Beam in space: On the intersection of the surface with the 1- and 2-axes (points 3, 7, 11, and 15 above for default integration).

Temperature and Field Variable Input at Specific Points for Beam Sections Integrated during the Analysis

Give the value at each of the points shown below.



Beam in a plane

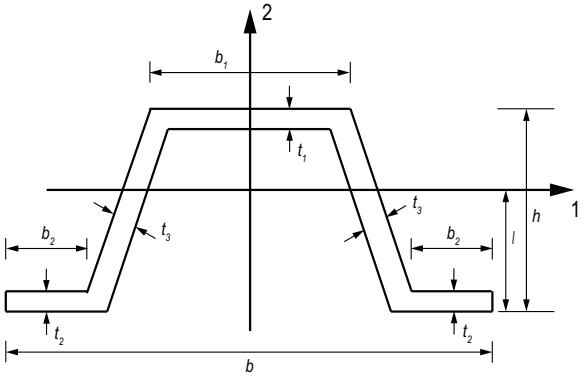


Beam in space

Temperature Input for a Frame Section

Constant temperature throughout the element cross-section is assumed; therefore, only one temperature value per node is required.

Hat Section



Input File Usage: Use the following option:

**BEAM GENERAL SECTION*, SECTION=HAT

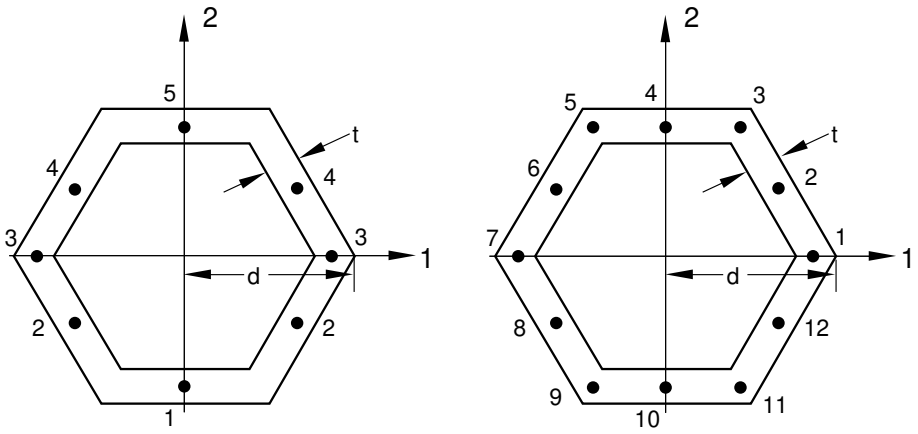
Abaqus/CAE Usage: Defining a hat section is not supported in Abaqus/CAE.

Geometric Input Data

l, h, b, $\boldsymbol{b_1}$, $\boldsymbol{b_2}$, $\boldsymbol{t_1}$, $\boldsymbol{t_2}$, $\boldsymbol{t_3}$

By allowing you to specify l, the origin of the local cross-section axis can be placed anywhere on the symmetry line (the local 2-axis). In the above figure a negative value of l implies that the origin of the local cross-section axis is below the lower edge of the bottom flanges, which may be needed when constraining a beam stiffener to a shell.

Hexagonal Section



Default integration, beam in a plane

Default integration, beam in space

Input File Usage: Use either of the following options:

**BEAM SECTION*, SECTION=HEX
**BEAM GENERAL SECTION*, SECTION=HEX

Abaqus/CAE Usage: Property module: **Create Profile**: **Hexagonal**

Geometric Input Data

d (circumscribing radius), t (wall thickness)

Default Integration (Simpson)

Beam in a plane: 5 points

Beam in space: 3 points in each wall segment (12 total)

Nondefault Integration Input for a Beam Section Integrated during the Analysis

Beam in a plane: Give the number of points along the section wall, moving in the second beam section axis direction. This number must be odd and greater than or equal to three.

Beam in space: Give the number of points in each wall segment. This number must be odd and greater than or equal to three.

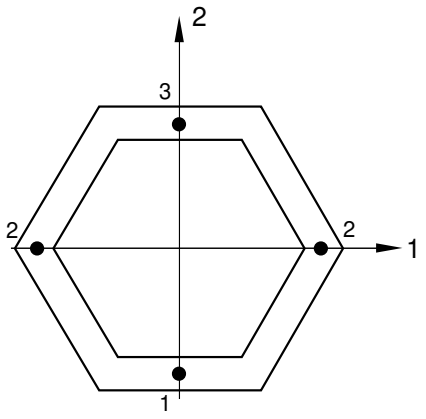
Default Stress Output Points If a Beam Section Integrated during the Analysis Is Used

Beam in a plane: Bottom and top (points 1 and 5 above for default integration).

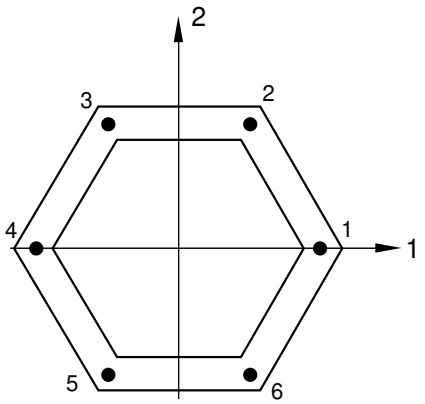
Beam in space: Vertices (points 1, 3, 5, 7, 9, and 11 above for default integration).

Temperature and Field Variable Input at Specific Points for Beam Sections Integrated during the Analysis

Give the value at each of the points shown below.

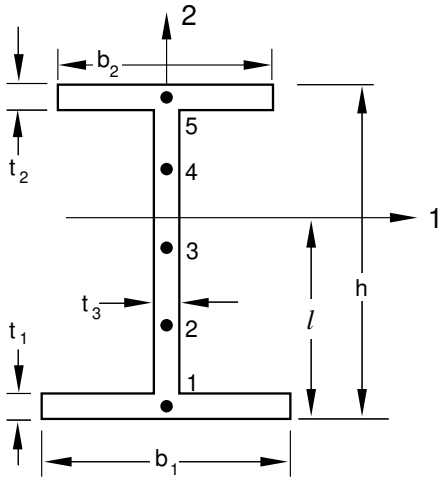


Beam in a plane

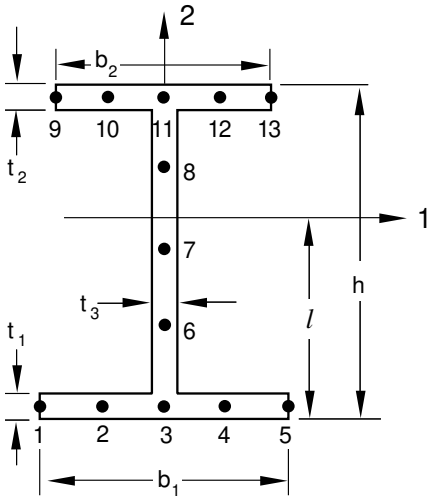


Beam in space

I-Section



Default integration, beam in a plane



Default integration, beam in space

Input File Usage: Use one of the following options:

**BEAM SECTION*, SECTION=I
**BEAM GENERAL SECTION*, SECTION=I
**FRAME SECTION*, SECTION=I

Abaqus/CAE Usage: Property module: **Create Profile**: **I**

Geometric Input Data

l, *h*, b_1, b_2, t_1, t_2, t_3

By allowing you to specify *l*, the origin of the local cross-section axis can be placed anywhere on the symmetry line (the local 2-axis). In the above figures a negative value of *l* implies that the origin of the local cross-section axis is below the lower edge of the bottom flange, which may be needed when constraining a beam stiffener to a shell.

Defining a T-Section

Input File Usage: Set b_1 and t_1 or b_2 and t_2 to zero to model a T-section.

Abaqus/CAE Usage: Property module: **Create Profile**: **T**

Default Integration (Simpson)

Beam in a plane: 5 points (one in each flange plus 3 in web)

Beam in space: 5 points in web, 5 in each flange (13 total)

Nondefault Integration Input for a Beam Section Integrated during the Analysis

Beam in a plane: Give the number of points in the second beam section axis direction. This number must be odd and greater than or equal to three.

Beam in space: Give the number of points in the lower flange first, then in the web, and then in the upper flange. These numbers must be odd and greater than or equal to three in each nonvanishing section.

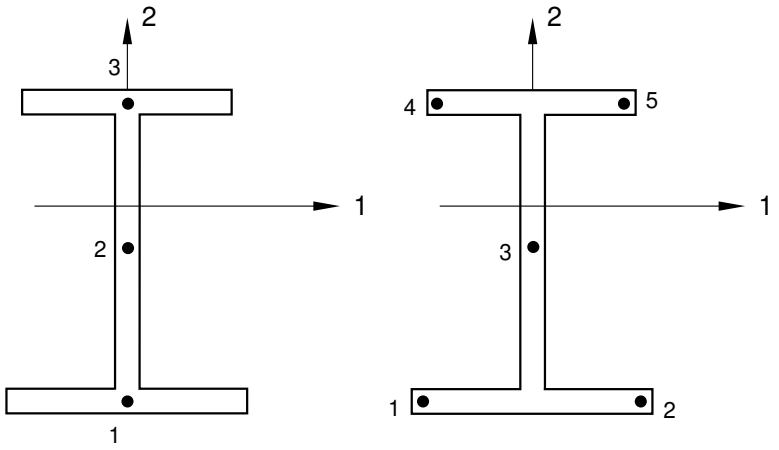
Default Stress Output Points If a Beam Section Integrated during the Analysis Is Used

Beam in a plane: Flanges (points 1 and 5 above for default integration).

Beam in space: Ends of flanges (points 1, 5, 9, and 13 above for default integration).

Temperature and Field Variable Input at Specific Points for Beam Sections Integrated during the Analysis

Give the value at each of the points shown below.



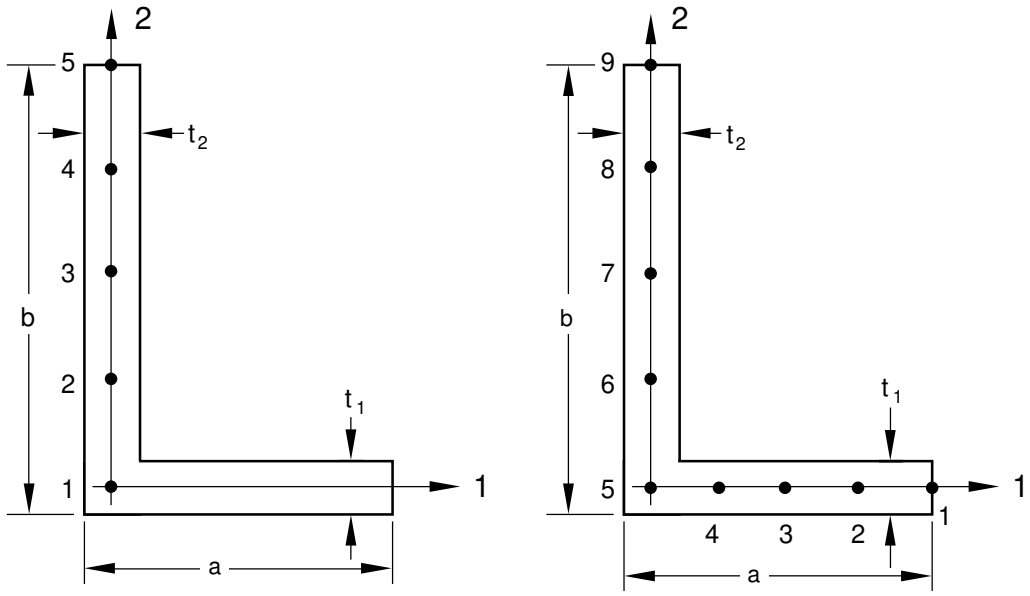
Beam in a plane Beam in space

For a beam in space the temperature is first interpolated linearly through the flanges based on the temperature at points 1 and 2, and then 4 and 5, respectively. It is then interpolated parabolically through the web.

Temperature Input for a Frame Section

Constant temperature throughout the element cross-section is assumed; therefore, only one temperature value per node is required.

L-Section



Default integration, beam in a plane

Default integration, beam in space

Input File Usage: Use either of the following options:

**BEAM SECTION*, SECTION=L
**BEAM GENERAL SECTION*, SECTION=L

Abaqus/CAE Usage: Property module: **Create Profile**: **L**

Geometric Input Data

a, b, t_1, t_2

Default Integration (Simpson)

Beam in a plane: 5 points

Beam in space: 5 points in each flange (9 total)

Nondefault Integration Input for a Beam Section Integrated during the Analysis

Beam in a plane: Give the number of points in the second beam section axis direction. This number must be odd and greater than or equal to three.

Beam in space: Give the number of points in the first beam section axis direction, then the number of points in the second beam section axis direction. These numbers must be odd and greater than or equal to three.

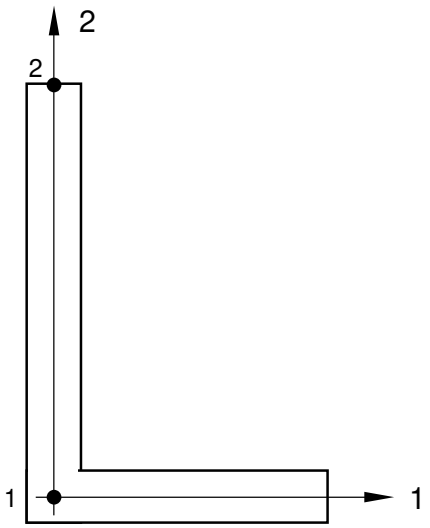
Default Stress Output Points If a Beam Section Integrated during the Analysis Is Used

Beam in a plane: Bottom and top (points 1 and 5 above for default integration).

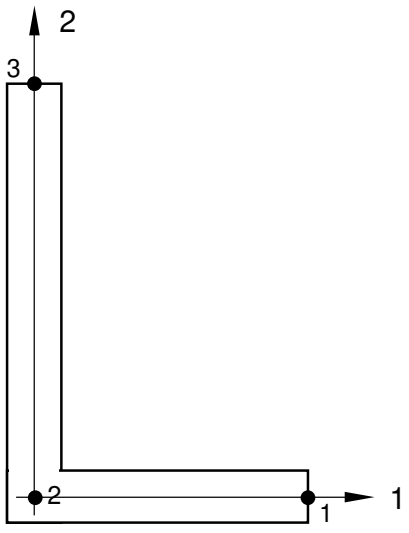
Beam in space: End of flange along positive local 1-axis; section corner; end of flange along positive local 2-axis (points 1, 5, and 9 above for default integration).

Temperature and Field Variable Input at Specific Points for Beam Sections Integrated during the Analysis

Give the value at each of the points shown below.

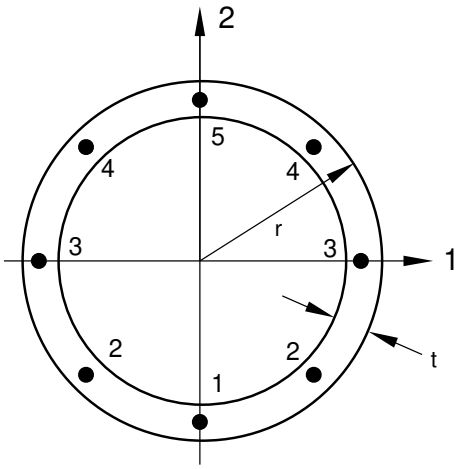


Beam in a plane

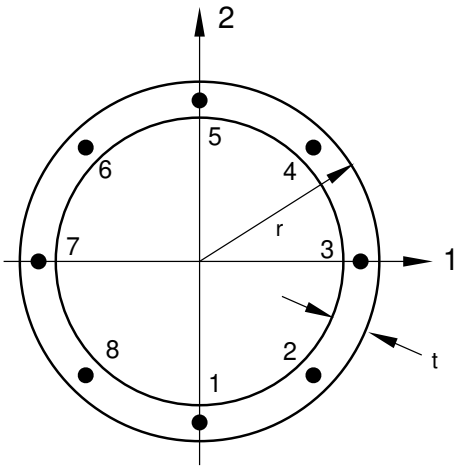


Beam in space

Pipe Section (Thin-Walled)



Default integration,
beam in a plane



Default integration,
beam in space

Pipe cross-sections can be associated with beam, pipe, or frame elements.

Input File Usage: Use one of the following options:

**BEAM SECTION*, SECTION=PIPE
**BEAM GENERAL SECTION*, SECTION=PIPE
**FRAME SECTION*, SECTION=PIPE

Abaqus/CAE Usage: Property module: **Create Profile**: **Pipe**: **Thin walled**

Geometric Input Data

r (outside radius), *t* (wall thickness)

Default Integration

Beam in a plane: 5 points (Simpson's rule)

Beam in space: 8 points (trapezoidal rule)

Nondefault Integration Input for a Beam Section Integrated during the Analysis

Beam in a plane: Give an odd number of points. This number must be greater than or equal to five.

Beam in space: Give an even number of points. This number must be greater than or equal to eight.

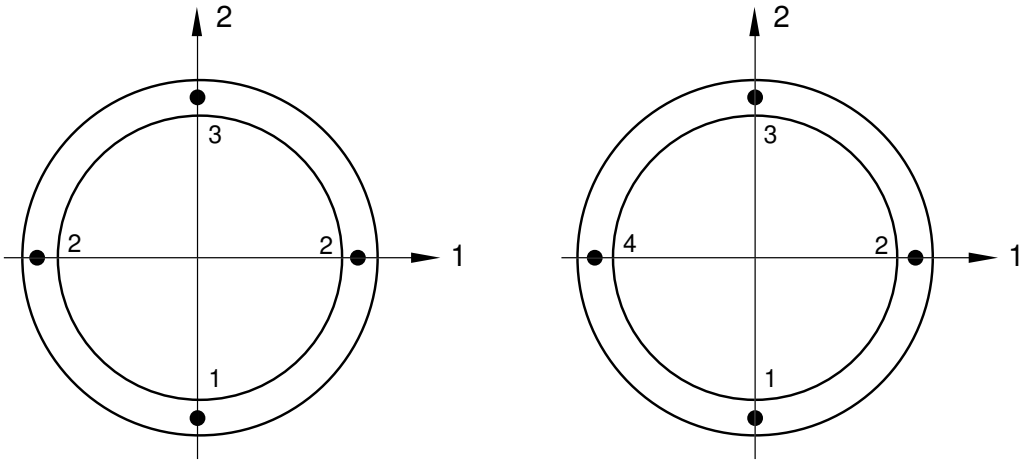
Default Stress Output Points If a Beam Section Integrated during the Analysis Is Used

Beam in a plane: Bottom and top (points 1 and 5 above for default integration).

Beam in space: On the intersection of the surface with the 1- and 2-axes (points 1, 3, 5, and 7 above for default integration).

Temperature and Field Variable Input at Specific Points for Beam Sections Integrated during the Analysis

Give the value at each of the points shown below.

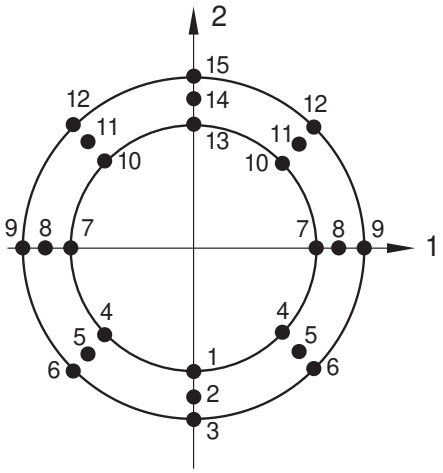


Beam in a plane Beam in space

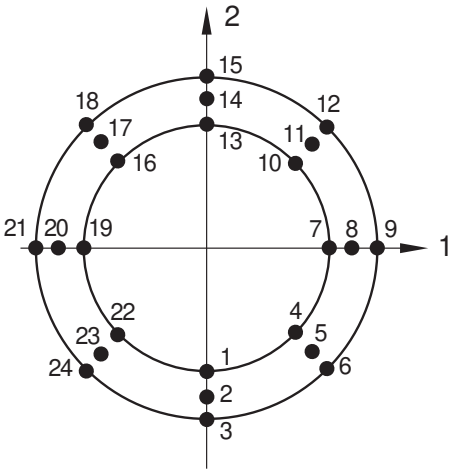
Temperature Input for a Frame Section

Constant temperature throughout the element cross-section is assumed; therefore, only one temperature value per node is required.

Pipe Section (Thick-Walled)



Default integration, beam in a plane



Default integration, beam in space

Thick-walled pipe cross-sections can be associated with beam or pipe elements.

Input File Usage: Use the following option:

**BEAM SECTION*, SECTION=THICK PIPE

Abaqus/CAE Usage: Property module: **Create Profile**: **Pipe**: **Thick walled**

Geometric Input Data

r (outside radius), *t* (wall thickness)

Default Integration

Beam in a plane: 3 points radially (Simpson's rule), 5 circumferentially (trapezoidal rule)

Beam in space: 3 points radially (Simpson's rule), 8 circumferentially (trapezoidal rule)

Nondefault Integration Input for a Beam Section Integrated during the Analysis

Beam in a plane: Give an odd number of points in the radial direction, then an odd number of points (greater than or equal to 5) in the circumferential direction.

Beam in space: Give an odd number of points in the radial direction, then an even number of points (greater than or equal to 8) in the circumferential direction.

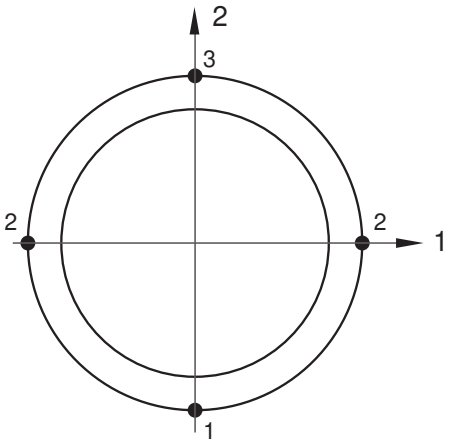
Default Stress Output Points If a Beam Section Integrated during the Analysis Is Used

Beam in a plane: Bottom and top on the pipe midsurface (points 2 and 14 above for default integration).

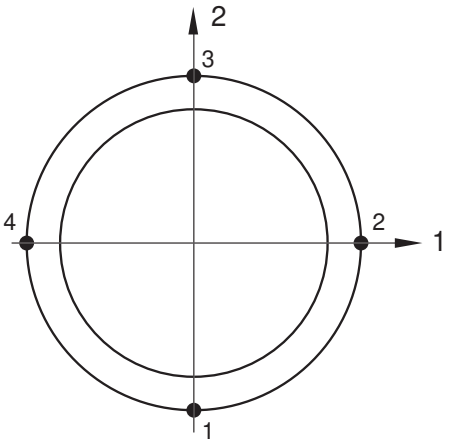
Beam in space: On the intersection of the pipe midsurface with the 1- and 2-axes (points 2, 8, 14, and 20 above for default integration).

Temperature and Field Variable Input at Specific Points for Beam Sections Integrated during the Analysis

Give the value at each of the points shown below.

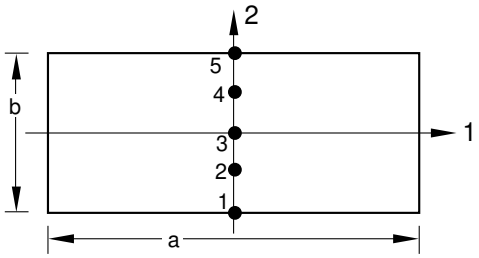


Beam in a plane

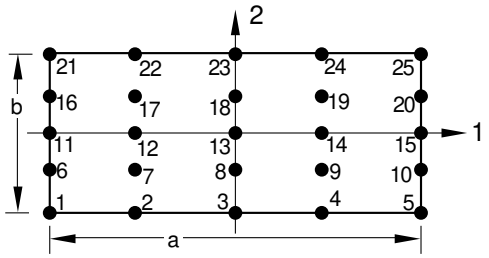


Beam in space

Rectangular Section



Default integration, beam in a plane



Default integration, beam in space

Input File Usage: Use one of the following options:

**BEAM SECTION*, SECTION=RECT
**BEAM GENERAL SECTION*, SECTION=RECT
**FRAME SECTION*, SECTION=RECT

Abaqus/CAE Usage: Property module: **Create Profile**: **Rectangular**

Geometric Input Data

a, *b*

Default Integration (Simpson)

Beam in a plane: 5 points

Beam in space: 5×5 (25 total)

Nondefault Integration Input for a Beam Section Integrated during the Analysis

Beam in a plane: Give the number of points in the second beam section axis direction. This number must be odd and greater than or equal to five.

Beam in space: Give the number of points in the first beam section axis direction, then the number of points in the second beam section axis direction. These numbers must be odd and greater than or equal to five.

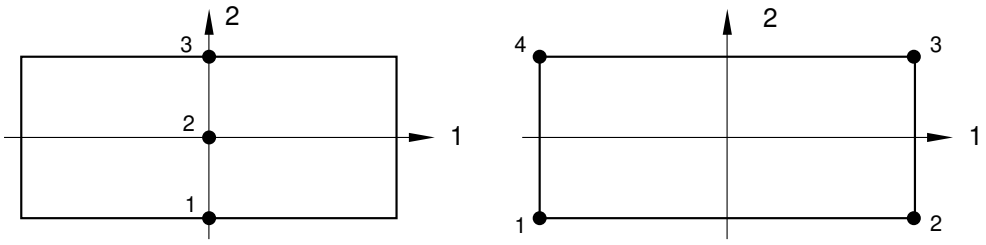
Default Stress Output Points If a Beam Section Integrated during the Analysis Is Used

Beam in a plane: Bottom and top (points 1 and 5 above for default integration).

Beam in space: Corners (points 1, 5, 21, and 25 above for default integration).

Temperature and Field Variable Input at Specific Points for Beam Sections Integrated during the Analysis

Give the value at each of the points shown below.

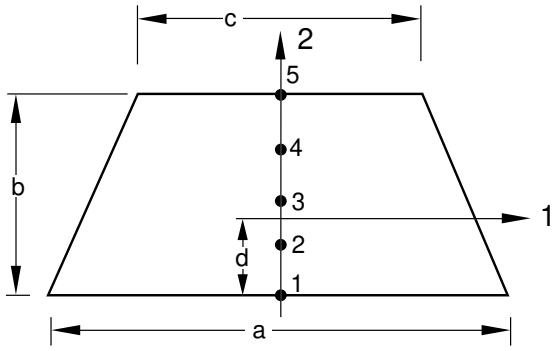


Beam in a plane Beam in space

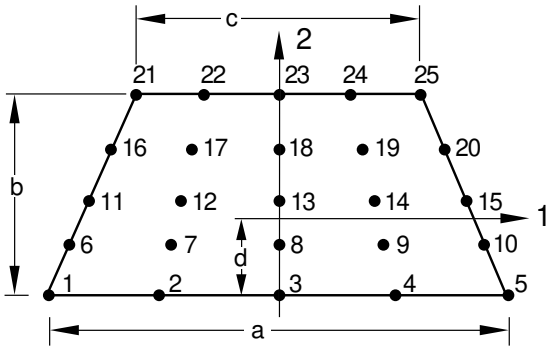
Temperature Input for a Frame Section

Constant temperature throughout the element cross-section is assumed; therefore, only one temperature value per node is required.

Trapezoidal Section



Default integration,
beam in a plane



Default integration,
beam in space

Input File Usage: Use either of the following options:

**BEAM SECTION*, SECTION=TRAPEZOID
**BEAM GENERAL SECTION*, SECTION=TRAPEZOID

Abaqus/CAE Usage: Property module: **Create Profile**: **Trapezoidal**

Geometric Input Data

a, *b*, *c*, *d*

By allowing you to specify *d*, the origin of the local cross-section axes can be placed anywhere on the symmetry line (the local 2-axis). In the above figures a negative value of *d* implies that the origin of the local cross-section axis is below the lower edge of the section. This may be needed when constraining a beam stiffener to a shell.

Default Integration (Simpson)

Beam in a plane: 5 points

Beam in space: 5 × 5 (25 total)

Nondefault Integration Input for a Beam Section Integrated during the Analysis

Beam in a plane: Give the number of points in the second beam section axis direction. This number must be odd and greater than or equal to five.

Beam in space: Give the number of points in the first beam section axis direction, then the number of points in the second beam section axis direction. These numbers must be odd and greater than or equal to five.

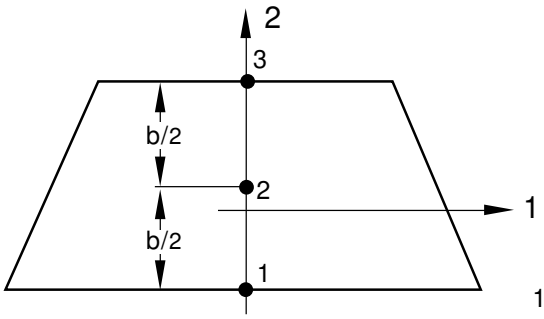
Default Stress Output Points If a Beam Section Integrated during the Analysis Is Used

Beam in a plane: Bottom and top (points 1 and 5 above for default integration).

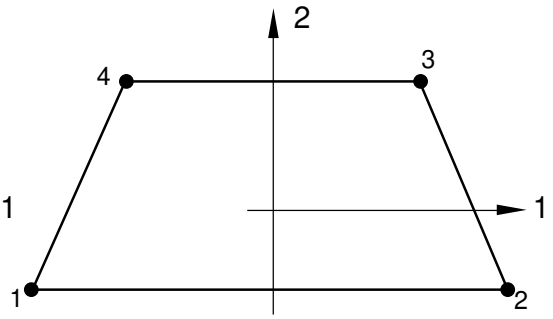
Beam in space: Corners (points 1, 5, 21, and 25 above for default integration).

Temperature and Field Variable Input at Specific Points for Beam Sections Integrated during the Analysis

Give the value at each of the points shown below.



Beam in a plane



Beam in space

Frame Elements

In this section:

- *Frame Elements*
- *Frame Section Behavior*
- *Frame Element Library*

Frame Elements

Products: Abaqus/Standard

References:

- *About Beam Modeling*
- *Frame Section Behavior*
- *Frame Element Library*
- **FRAME SECTION*

Overview

Frame elements:

- are 2-node, initially straight, slender beam elements intended for use in the elastic or elastic-plastic analysis of frame-like structures;
- are available in two or three dimensions;
- have elastic response that follows Euler-Bernoulli beam theory with fourth-order interpolation for the transverse displacements;
- have plastic response that is concentrated at the element ends (plastic hinges) and is modeled with a lumped plasticity model that includes nonlinear kinematic hardening;
- are implemented for small or large displacements (large rotations with small strains);
- output forces and moments at the element ends and midpoint;
- output elastic axial strain and curvatures at the element ends and midpoint and plastic displacements and rotations at the element ends only;
- admit, optionally, a uniaxial “buckling strut” response where the axial response of the element is governed by a damaged elasticity model in compression and an isotropic hardening plasticity model in tension and where all transverse forces and moments are zero;
- can switch to buckling strut response during the analysis (for pipe sections only); and
- can be used in static, implicit dynamic, and eigenfrequency extraction analyses only.

Typical Applications

Frame elements are designed to be used for small-strain elastic or elastic-plastic analysis of frame-like structures composed of slender, initially straight beams. Typically, a single frame element will represent the entire structural member connecting two joints. A frame element's elastic response is governed by Euler-Bernoulli beam theory with fourth-order interpolations for the transverse displacement field; hence, the element's kinematics include the exact (Euler-Bernoulli) solution to concentrated end forces and moments and constant distributed loads. The elements can be used to solve a wide variety of civil engineering design applications, such as truss structures, bridges, internal frame structures of buildings, off-shore platforms, and jackets, etc. A frame element's plastic response is modeled with a lumped plasticity model at the element ends that simulates the formation of plastic hinges. The lumped plasticity model includes nonlinear kinematic hardening. The elements can, thus, be used for collapse load prediction based on the formation of plastic hinges.

Slender, frame-like members loaded in compression often buckle in such a way that only axial force is supported by the member; all other forces and moments are negligibly small. Frame elements offer optional buckling strut response whereby the element only carries axial force, which is calculated based on a damaged elasticity model in compression and an isotropic hardening plasticity model in tension. This model provides a simple

phenomenological approximation to the highly nonlinear geometric and material response that takes place during buckling and postbuckling deformation of slender members loaded in compression.

For pipe sections only, frame elements allow switching to optional uniaxial buckling strut response during the analysis. The criterion for switching is the "ISO" equation together with the "strength" equation (see *Buckling strut response for frame elements*). When the ISO and strength equations are satisfied, the elastic or elastic-plastic frame element undergoes a one-time-only switch in behavior to buckling strut response.

Element Cross-Sectional Axis System

The orientation of the frame element's cross-section is defined in Abaqus/Standard in terms of a local, right-handed ($\mathbf{t}$, $\mathbf{n_1}$, $\mathbf{n_2}$) axis system, where $\mathbf{t}$ is the tangent to the axis of the element, positive in the direction from the first to the second node of the element, and $\mathbf{n_1}$ and $\mathbf{n_2}$ are basis vectors that define the local 1- and 2-directions of the cross-section. $\mathbf{n_1}$ is referred to as the first axis direction, and $\mathbf{n_2}$ is referred to as the normal to the element. Since these elements are initially straight and assume small strains, the cross-section directions are constant along each element and possibly discontinuous between elements.

Defining the N_1-Direction at the Nodes

For frame elements in a plane the $\mathbf{n_1}$-direction is always (0.0, 0.0, −1.0); that is, normal to the plane in which the motion occurs. Therefore, planar frame elements can bend only about the first axis direction.

For frame elements in space the approximate direction of $\mathbf{n_1}$ must be defined directly as part of the element section definition or by specifying an additional node off the element's axis. This additional node is included in the element's connectivity list (see *Element Definition*).

- If an additional node is specified, the approximate direction of $\mathbf{n_1}$ is defined by the vector extending from the first node of the element to the additional node.
- If both input methods are used, the direction calculated by using the additional node will take precedence.
- If the approximate direction is not defined by either of the above methods, the default value is (0.0, 0.0, −1.0).

The $\mathbf{n_1}$-direction is then the normal to the element's axis that lies in the plane defined by the element's axis and this approximate $\mathbf{n_1}$-direction. The $\mathbf{n_2}$-direction is defined as $\mathbf{t} \times \mathbf{n_1}$.

Large-Displacement Assumptions

The frame element's formulation includes the effect of large rigid body motions (displacements and rotations) when geometrically nonlinear analysis is selected (see *General and Perturbation Procedures*). Strains in these elements are assumed to remain small.

Material Response (Section Properties) of Frame Elements

For frame elements the geometric and material properties are specified together as part of the frame section definition. No separate material definition is required. You can choose one of the section shapes that is valid for frame elements from the beam cross-section library (see *Beam Cross-Section Library*). The valid section shapes depend upon whether elastic or elastic-plastic material response is specified or whether buckling strut response is included. See *Frame Section Behavior* for a complete discussion of specifying the geometric and material section properties.

Input File Usage: **FRAME SECTION*, SECTION=*section_type*

Mechanical Response and Mass Formulation

The mechanical response of a frame element includes elastic and plastic behavior. Optionally, uniaxial buckling strut response is available.

Elastic Response

The elastic response of a frame element is governed by Euler-Bernoulli beam theory. The displacement interpolations for the deflections transverse to the frame element's axis (the local 1- and 2-directions in three dimensions; the local 2-direction in two dimensions) are fourth-order polynomials, allowing quadratic variation of the curvature along the element's axis. Thus, each single frame element exactly models the static, elastic solution to force and moment loading at its ends and constant distributed loading along its axis (such as gravity loading). The displacement interpolation along an element's axis is a second-order polynomial, allowing linear variation of the axial strain. In three dimensions the twist rotation interpolation along an element's axis is linear, allowing constant twist strain. The elastic stiffness matrix is integrated numerically and used to calculate 15 nodal forces and moments in three dimensions: an axial force, two shear forces, two bending moments, and a twist moment at each end node, and an axial force and two shear forces at the midpoint node. In two dimensions 8 nodal forces and moments exist: an axial force, a shear force, and a moment at each end, and an axial force and a shear force at the midpoint. The forces and moments are illustrated in *Figure 1*.

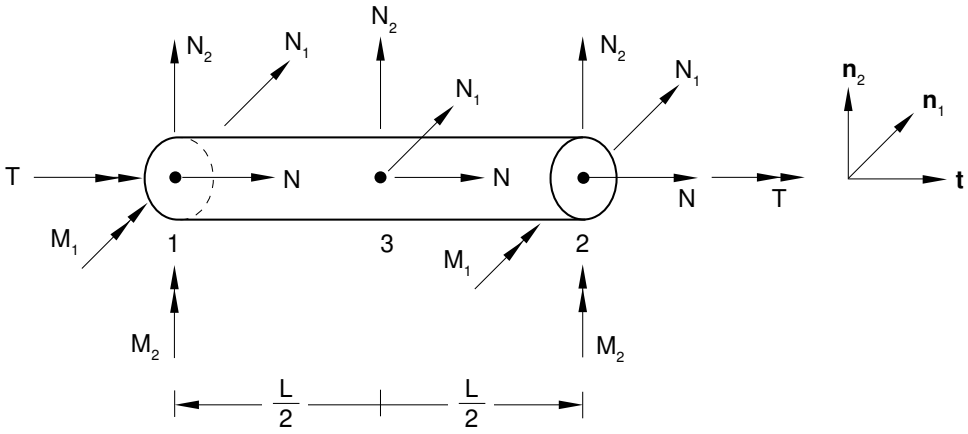


Figure 1: Forces and moments on a frame element in space.

Elastic-Plastic Response

The plastic response of the element is treated with a “lumped” plasticity model such that plastic deformations can develop only at the element's ends through plastic rotations (hinges) and plastic axial displacement. The growth of the plastic zone through the element's cross-section from initial yield to a fully yielded plastic hinge is modeled with nonlinear kinematic hardening. It is assumed that the plastic deformation at an end node is influenced by the moments and axial force at that node only. Hence, the yield function at each node, also called the plastic interaction surface, is assumed to be a function of that node's axial force and three moment components only. No length is associated with the plastic hinge. In reality, the plastic hinge will have a finite size determined by the element's length and the specific loading that causes yielding; the hinge size will influence the hardening rate but not the ultimate load. Hence, if the rate of hardening and, thus, the plastic deformation for a given load are important, the lumped plasticity model should be calibrated with the element's length and the loading situation taken into account. For details on the elastic-plastic element formulation, see *Frame elements with lumped plasticity*.

Uniaxial Linear Elastic and Buckling Strut Response with Tensile Yield

You can obtain a frame element's response to uniaxial force only, based on linear elasticity, buckling strut response, and tensile yield. In that case all transverse forces and moments in the element are zero. For linear

elastic response the element behaves like an axial spring with constant stiffness. For buckling strut response if the tensile axial force in the element does not exceed the yield force, the axial force in the element is constrained to remain inside a buckling envelope. See *Frame Section Behavior* for a description of this envelope. Inside the envelope the force is related to strain by a damaged elastic modulus. The cyclic, hysteretic response of this model is phenomenological and approximates the response of thin-walled, pipe-like members. When the element is loaded in tension beyond the yield force, the force response is governed by isotropic hardening plasticity. In reverse loading the response is governed by the buckling envelope translated along the strain axis by an amount equal to the axial plastic strain. For details of the buckling strut formulation, see *Buckling strut response for frame elements*.

Mass Formulation

The frame element uses a lumped mass formulation for both dynamic analysis and gravity loading. The mass matrix for the translational degrees of freedom is derived from a quadratic interpolation of the axial and transverse displacement components. The rotary inertia for the element is isotropic and concentrated at the two ends.

For buckling strut response a lumped mass scheme is used, where the element's mass is concentrated at the two ends; no rotary inertia is included.

Using Frame Elements in Contact Problems

When contact conditions play a role in a structure's behavior, frame elements have to be used with caution. A frame element has one additional internal node, located in the middle of the element. No contact constraint is imposed on this node, so this internal node may penetrate the surface in contact, resulting in a sagging effect.

Output

The forces and moments, elastic strains, and plastic displacements and rotations in a frame element are reported relative to a corotational coordinate system. The local coordinate directions are the axial direction and the two cross-sectional directions. Output of section forces and moments as well as elastic strains and curvatures is available at the element ends and midpoint. Output of plastic displacement and rotations is available only at the element ends. You can request output to the output database (at the integration points only), to the data file, or to the results file (see *Output to the Data and Results Files* and *Frame Elements*). Since frame elements are formulated in terms of section properties, stress output is not available.

Frame Section Behavior

Products: Abaqus/Standard

References:

- *Frame Elements*
- **FRAME SECTION*

Overview

The frame section behavior:

- requires definition of the section's shape and its material response;
- uses linear elastic behavior in the interior of the frame element;
- can include “lumped” plasticity at the element ends to model the formation of plastic hinges;
- can be uniaxial only, with response governed by a phenomenological buckling strut model, together with linear elasticity and tensile plastic yielding; and
- for pipe sections only, can switch to buckling strut response during the analysis.

Defining Elastic Section Behavior

The elastic response of the frame elements is formulated in terms of Young's modulus, E; the torsional shear modulus, G; coefficient of thermal expansion, α; and cross-section shape. Geometric properties such as the cross-sectional area, A, or bending moments of inertia are constant along the element and during the analysis.

If present, thermal strains are constant over the cross-section, which is equivalent to assuming that the temperature does not vary in the cross-section. As a result of this assumption only the axial force, N, depends on the thermal strain

$$N = EA\left(\varepsilon - \varepsilon^{th}\right),$$

where ε defines the total axial strain, including any initial elastic strain caused by a user-defined nonzero initial axial force, and ε^{th} defines the thermal expansion strain given by

$$\varepsilon^{th} = \alpha\left(\theta, f_\beta\right)\left(\theta - \theta^0\right) - \alpha\left(\theta^I, f_\beta{}^I\right)\left(\theta^I - \theta^0\right),$$

where

$\alpha(\theta)$

is the thermal expansion coefficient,

θ

is the current temperature at the section,

θ^0

is the reference temperature for α,

θ^I

is the user-defined initial temperature at this point (*Initial Conditions*),

f_β

are field variables, and

f_β^I

are the user-defined initial values of field variables at this point (*Initial Conditions*).

The bending moment and twist torque responses are defined by the constitutive relations

$$\begin{aligned} M_1 &= E\left(I_{11}\kappa_1 - I_{12}\kappa_2\right), \\ M_2 &= E\left(-I_{12}\kappa_1 + I_{22}\kappa_2\right), \\ T &= GJ\phi, \end{aligned}$$

where

I_{11}

is the moment of inertia for bending about the 1-axis of the section,

I_{22}

is the moment of inertia for bending about the 2-axis of the section,

I_{12}

is the moment of inertia for cross-bending,

J

is the torsional constant,

κ_1

is the curvature change about the first beam section local axis, including any elastic curvature change associated with a user-defined nonzero initial moment M_1 (*Initial Conditions*),

κ_2

is the curvature change about the second beam section local axis, including any elastic curvature change associated with a user-defined nonzero initial moment M_2 (*Initial Conditions*), and

ϕ

is the twist, including any elastic twist associated with a user-defined nonzero initial twisting moment (torque) T (*Initial Conditions*).

Defining Temperature and Field-Variable-Dependent Section Properties

The temperature and predefined field variables may vary linearly over the element's length. Material constants such as Young's modulus, $E(\theta, f_\beta)$, the torsional shear modulus, $G(\theta, f_\beta)$, and the coefficient of thermal expansion, $\alpha(\theta, f_\beta)$, can also depend on the temperature, θ, and field variables f_β. You must associate the section definition with an element set.

Input File Usage: **FRAME SECTION*, ELSET=*name*

Specifying a Standard Library Section and Allowing Abaqus/Standard to Calculate the Cross-Section's Parameters

Select one of the following section profiles from the standard library of cross-sections (see *Beam Cross-Section Library*): box, circular, I, pipe, or rectangular. Specify the geometric input data needed to define the shape of the cross-section. Abaqus/Standard will then calculate the geometric quantities needed to define the section behavior automatically.

Input File Usage: **FRAME SECTION*, SECTION=*library_section*, ELSET=*name*

Specifying the Geometric Quantities Directly

Specify a general cross-section to define the area of the cross-section, moments of inertia, and torsional constant directly. These data are sufficient for defining the elastic section behavior since the axial stretching, bending response, and torsional behavior are assumed to be uncoupled.

Input File Usage: **FRAME SECTION*, SECTION=GENERAL, ELSET=*name*

Specifying the Elastic Behavior

Specify the elastic modulus, the torsional shear modulus, and the coefficient of thermal expansion as functions of temperature and field variables.

Input File Usage: **FRAME SECTION*, SECTION=*section_type*, ELSET=*name*
first_data_line
second_data_line
elastic_modulus, *torsional_shear_modulus*, *coefficient_of_thermal_expansion*, *temperature*, *fv_1*, *fv_2*, etc.

Defining Elastic-Plastic Section Behavior

To include elastic-plastic response, specify N, M_1, M_2, and T directly as functions of their conjugate plastic deformation variables or use the default plastic response for N, M_1, M_2, and T based on the material yield stress. Abaqus/Standard uses the specified or default values to define a nonlinear kinematic hardening model that is "lumped" into plastic hinges at the element ends. Since the plasticity is lumped at the element ends, no length dimension is associated with the hinge. Generalized forces are related to generalized plastic displacements, not strains. In reality, the plastic hinge will have a finite size determined by the structural member's length and the loading, which will affect the hardening rate but not the ultimate load. For example, yielding under pure bending (a constant moment over the member) will produce a hinge length equal to the member length, whereas yielding of a cantilever with transverse tip load (a linearly varying moment over the member) will produce a much more localized hinge. Hence, if the rate of hardening and, thus, the plastic deformation at a given load are of importance, you should calibrate the plastic response appropriately for different lengths and different loading situations.

In the plastic range the only plastic surface available is an ellipsoid. This yield surface is only reasonably accurate for the pipe cross-section. Box, circular, I, and rectangular cross-sections can be used at your discretion with the understanding that the elliptic yield surface may not approximate the elastic-plastic response accurately. The general cross-section type cannot be used with plasticity.

Defining N, M_1, M_2, and T Directly

You can define N, M_1, M_2, and T directly. (See *Material Data Definition* for a detailed discussion of the tabular input conventions. In particular, you must ensure that the range of values given for the variables is sufficient for the application since Abaqus/Standard assumes a constant value of the dependent variable outside the specified range.) Abaqus/Standard will fit an exponential curve to the user-supplied data as discussed below (see "Elastic-plastic data curve fit and calculation of default values" below). The plastic data describe the response to axial force, moment about the cross-sectional 1- and 2-directions, and torque.

You must specify pairs of data relating the generalized force component to the appropriate plastic variable. Since the plasticity is concentrated at the element ends, the overall plastic response is dependent on the length of the element; hence, members with different lengths might require different hardening data. The plasticity model for frame elements is intended for frame-like structures: each member between structural joints is modeled with a single frame element where plastic hinges are allowed to develop at the end connections.

At least three data pairs for each plastic variable are required to describe the elastic-plastic section hardening behavior. If fewer than three data pairs are given, Abaqus/Standard will issue an error message.

Input File Usage: Use the following options:

**FRAME SECTION*, SECTION=PIPE, ELSET=*name*
**PLASTIC AXIAL* for N
**PLASTIC M1* for M_1
**PLASTIC M2* for M_2
**PLASTIC TORQUE* for T

Allowing Abaqus/Standard to Calculate Default Values for N, M_1, M_2, and T

You can use the default elastic-plastic material response for the plastic variables based on the yield stress for the material. The default elastic-plastic material response differs for each of the plastic variables: the plastic axial force, first plastic bending moment, second plastic bending moment, and plastic torsional moment. Specific default values are given below.

If you define the plastic variables directly and specify that the default response should be used, the data defined by you will take precedence over the default values.

Input File Usage: Use the following options:

**FRAME SECTION*, SECTION=PIPE, ELSET=*name*,
PLASTIC DEFAULTS, YIELD STRESS=σ^0
plastic options if user-defined values are necessary for a particular generalized force

Elastic-Plastic Data Curve Fit and Calculation of Default Values

The elastic-plastic response is a nonlinear kinematic hardening plasticity model. See *Models for Metals Subjected to Cyclic Loading* for a discussion of the nonlinear kinematic hardening formulation.

Nonlinear Kinematic Hardening with N, M_1, M_2, and T Defined Directly

For each of the four plastic material variables Abaqus/Standard uses an exponential curve fit of the user-supplied generalized force versus generalized plastic displacement to define the limits on the elastic range. The curve-fit procedure generates a hardening curve from the user-supplied data. It requires at least three data pairs.

The nonlinear kinematic hardening model describes the translation of the yield surface in generalized force space through a generalized backstress, $\boldsymbol{\alpha}$. The kinematic hardening is defined to be an additive combination of a purely kinematic linear hardening term and a relaxation (recall) term such that the backstress evolution is defined by

$$\dot{\boldsymbol{\alpha}} = \left(\operatorname{sign}\left(F - \boldsymbol{\alpha}\right) C - \gamma\boldsymbol{\alpha}\right) \left|\dot{q}^{pl}\right|,$$

where F is a component of generalized force, and C and $\boldsymbol{\gamma}$ are material parameters that are calibrated based on the user-defined or default hardening data. C is the initial hardening modulus, and $\boldsymbol{\gamma}$ determines the rate at which the kinematic hardening modulus decreases with increasing backstress, $\boldsymbol{\alpha}$. The saturation value of $\boldsymbol{\alpha}$ ($\dot{\boldsymbol{\alpha}} = \mathbf{0}$), called $\boldsymbol{\alpha}^{\boldsymbol{s}}$, is

$$\boldsymbol{\alpha}^{\boldsymbol{s}} = \frac{C}{\gamma}.$$

See *Figure 1* for an illustration of the elastic range for the nonlinear kinematic hardening model.

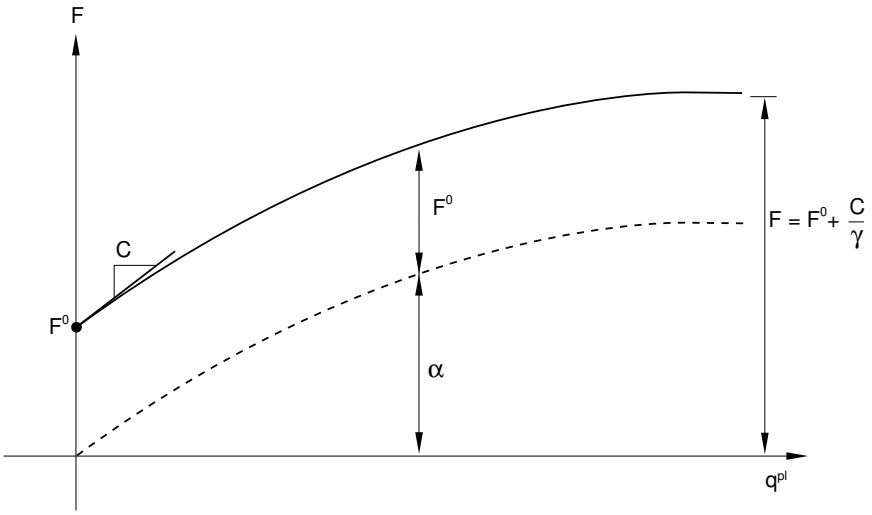


Figure 1: Nonlinear kinematic hardening model: yield surface for positive loading and the center of the yield surface, α.

Allowing Abaqus/Standard to Generate the Default Nonlinear Kinematic Hardening Model

To define the default plastic response, three data points are generated from the yield stress value and the cross-section shape. These three data points relate generalized force to generalized plastic displacement per unit length of the element. Since the model is calibrated per unit element length, the generated default plastic response is different for different element lengths. The generalized force levels for these three points are $\boldsymbol{F^0}$, $\boldsymbol{F^1}$, and $\boldsymbol{F^2}$. $\boldsymbol{F^0}$ is the generalized force at zero plastic generalized displacement. $\boldsymbol{F^1}$ and $\boldsymbol{F^2}$ are generalized force magnitudes that characterize the ultimate load-carrying capacity. The slopes between the data points (i.e., the generalized plastic moduli $\boldsymbol{D_1}$ and $\boldsymbol{D_2}$) characterize the hardening response. See *Figure 2* for an illustration of the default nonlinear kinematic hardening model.

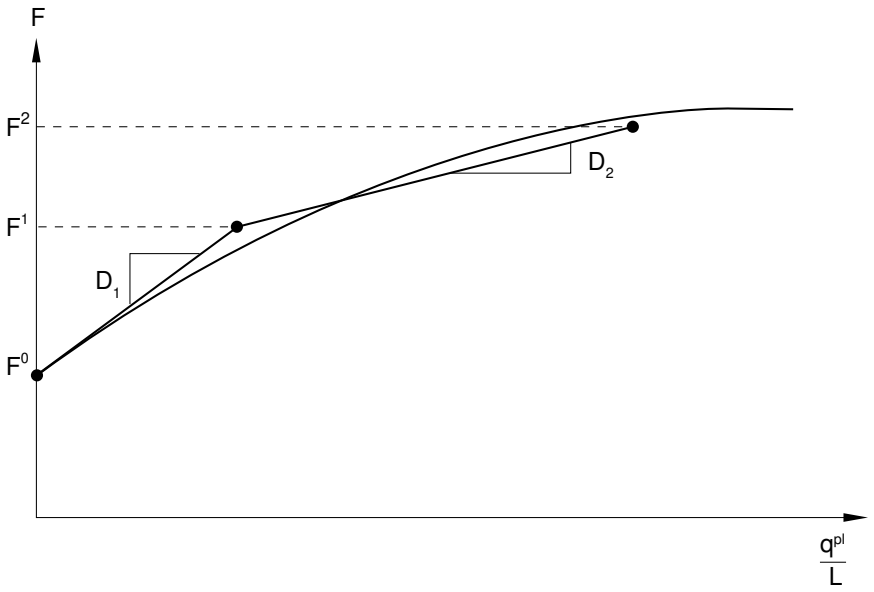


Figure 2: Data points generated for the default nonlinear kinematic hardening model.

For the plastic axial force, F^0 is the axial force that causes initial yielding. For the plastic bending moments about the first and second axes, F^0 is the moment about the first and second cross-sectional directions, respectively, that produces first fiber yielding. For the plastic torsional moment, F^0 is the torque about the axis that produces first fiber yielding. The generalized force levels F^1 and F^2, along with the connecting slopes D_1 and D_2, are chosen to approximate the response of a pipe cross-section made of a typical structural steel, with mild work hardening, from initial yielding to the development of a fully plastic hinge. The work hardening of the material corresponds to the default hardening of the section during axial loading. For different loading situations the size of the plastic hinge will vary; hence, the default model should be checked for validity against all anticipated loading situations. Default values for F^1, F^2, D_1, and D_2 corresponding to each plastic variable are listed in *Table 1*.

Table 1: Default values for generalized forces and connecting slopes for corresponding plastic variables.

	F^1	D_1	F^2	D_2
Plastic axial force	$1.05F^0$	$0.02EA$	$1.075F^0$	$0.01EA$
First plastic bending moment	a_3F^0	a_1EI_{11}	$1.1a_3F^0$	a_2EI_{11}
Second plastic bending moment	a_3F^0	a_1EI_{22}	$1.1a_3F^0$	a_2EI_{22}
Plastic torsional moment (for box and pipe sections)	a_3F^0	a_1GJ	$1.445F^0$	a_2GJ
Plastic torsional moment (for I-sections)	$1.31F^0$	$0.265EA$	$1.445F^0$	$0.06EA$

These default values are available for pipe, box, and I cross-section types with the values for the coefficients a_1, a_2, and a_3 as shown in *Table 2*.

Table 2: Coefficients a_1, a_2, and a_3.

Cross-section type	a_1	a_2	a_3
Pipe	0.30	0.07	1.35
Box	0.17	0.02	1.20
I (strong)	0.10	0.02	1.12
I (weak)	0.43	0.10	1.50

Defining Optional Uniaxial Strut Behavior

Frame elements optionally allow only uniaxial response (strut behavior). In this case neither end of the element supports moments or forces transverse to the axis; hence, only a force along the axis of the element exists. Furthermore, this axial force is constant along the length of the element, even if a distributed load is applied tangentially to the element axis. The uniaxial response of the element is linear elastic or nonlinear, in which case it includes buckling and postbuckling in compression and isotropic hardening plasticity in tension.

Defining Linear Elastic Uniaxial Behavior

A linear elastic uniaxial frame element behaves like an axial spring with constant stiffness $\boldsymbol{EA/L}$, where E is Young's modulus, A is the cross-sectional area, and L is the original element length. The strain measure is the change in length of the element divided by the element's original length.

Input File Usage: **FRAME SECTION*, SECTION=*library_section*, ELSET=*name*, PINNED

Defining Buckling, Postbuckling, and Plastic Uniaxial Behavior: Buckling Strut Response

If uniaxial buckling and postbuckling in compression and isotropic hardening plasticity in tension are modeled (buckling strut response), the buckling envelope must be defined. The buckling envelope defines the force versus axial strain (change in length divided by the original length) response of the element. It is illustrated in *Figure 3*.

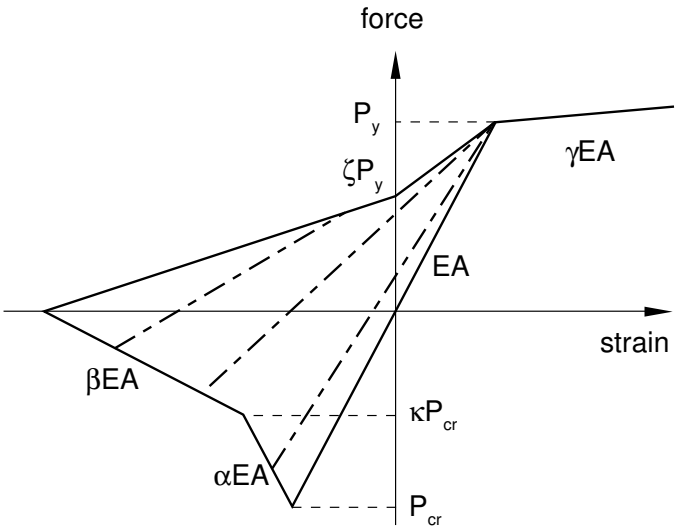


Figure 3: Buckling envelope for uniaxial buckling response.

The buckling envelope derives from Marshall Strut theory, which is developed for pipe cross-section profiles only. No other cross-section types are permitted with buckling strut response.

Seven coefficients determine the buckling envelope as follows (the default values are listed, where D is the pipe outer diameter and t is the pipe wall thickness):

$P_y = \xi \sigma^0 A$

Elastic limit force ($\xi = 0.95$). σ^0 is the yield stress.

γEA

Isotropic hardening slope ($\gamma = 0.02$).

P_{cr}

Critical compressive buckling force predicted by the ISO equation, defined in *Buckling strut response for frame elements*.

αEA

Slope of a segment on the buckling envelope, $\alpha = \alpha_0 + \alpha_1 \frac{L}{D}$ ($\alpha_0 = 0.03$ and $\alpha_1 = 0.004$).

κP_{cr}

Corner on the buckling envelope ($\kappa = 0.28$).

βEA

Slope of a segment on the buckling envelope ($\beta = 0.02$).

ζP_y

Corner on the buckling envelope ($\zeta = min\left(1.0, \frac{5.8}{\xi}\left(\frac{t}{D}\right)^{0.7}\right)$).

The axial force in the element is required to stay inside or on the buckling envelope. When tension yielding occurs, the enclosed part of the envelope translates along the strain axis by an amount equal to the plastic strain. When reverse loading occurs for points on the boundary of the enclosed part of the envelope, the strut exhibits "damaged elastic" behavior. This damaged elastic response is determined by drawing a line from the point on the envelope to the tension yield point (force value P_y). As long as the force and axial strain remain inside the enclosed part of the envelope, the force response is linear elastic with a modulus equal to the damaged elastic modulus. At any time that the compressive strain is greater in magnitude than the negative extreme strain point of the envelope, the force is constant with a value of zero.

The value of P_{cr} is a function of an element's geometrical and material properties, including the yield stress value.

Buckling strut response cannot be used with elastic-plastic frame section behavior; the strut's plastic behavior is defined by P_y and the isotropic hardening slope γEA.

Defining the Buckling Envelope

You can specify that the default buckling envelope should be used, or you can define the buckling envelope. If you define the buckling envelope directly and specify that the default envelope should be used, the values defined by you will take precedence.

In either case you must provide the yield stress value, which will be used to determine the yield force in tension and the critical compressive buckling load (through the ISO equation described later in this section).

Input File Usage: To specify the default buckling envelope, use the following option:

**FRAME SECTION*, SECTION=PIPE, ELSET=*name*, BUCKLING,
PINNED, YIELD STRESS=σ^0

To specify a user-defined buckling envelope, use both of the following options:

**FRAME SECTION*, SECTION=PIPE, ELSET=*name*, PINNED, YIELD STRESS=σ^0

**BUCKLING ENVELOPE*

Defining the Critical Buckling Load

The critical buckling load, P_{cr}, is determined by the ISO equation, which is an empirical relationship determined by the International Organization for Standardization based on experimental results for pipe-like or tubular structural members. Within the ISO equation, four variables can be changed from their default values: the effective length factors, k_1 and k_2, in the first and second sectional directions (the default values are 1.0) and the added length, ΔL_1 and ΔL_2 in the first and second sectional directions (the default values are 0). These variables account for the buckling member's end connectivity. The effective element length in the transverse direction i ($i = 1, 2$) is $\overline{L}_i = k_i (L + \Delta L_i)$. For details on the ISO equation, see *Buckling strut response for frame elements*.

Input File Usage: To define nondefault coefficients for the ISO equation with the default buckling envelope, use both of the following options:

**FRAME SECTION*, SECTION=PIPE, ELSET=*name*, BUCKLING,
PINNED, YIELD STRESS=σ^0
**BUCKLING LENGTH*

To define nondefault coefficients for the ISO equation with a user-defined buckling envelope, use all of the following options:

**FRAME SECTION*, SECTION=PIPE, ELSET=*name*, PINNED, YIELD STRESS=σ^0

**BUCKLING ENVELOPE*
**BUCKLING LENGTH*

Switching to Optional Uniaxial Strut Behavior during an Analysis

Frame elements allow switching to uniaxial buckling strut response during the analysis. The criterion for switching is the "ISO" equation together with the "strength" equation (see *Buckling strut response for frame elements*). When the ISO equation is satisfied, the elastic or elastic-plastic frame element undergoes a one-time-only switch in behavior to buckling strut response. The strength equation is introduced to prevent switching in the absence of significant axial forces.

When the frame element switches to buckling strut response, a dramatic loss of structural stiffness occurs. The switched element no longer supports bending, torsion, or shear loading. If the global structure is unstable as a result of the switch (that is, the structure would collapse under the applied loading), the analysis may fail to converge.

To permit switching of the element response, use the default buckling envelope or define a buckling envelope and provide a yield stress, but do not activate linear elastic uniaxial behavior for the frame element.

The ISO equation is an empirical relationship based on experiments with slender, pipe-like (tubular) members. Since the equation is written explicitly in terms of the pipe outer diameter and thickness, only pipe sections are permitted with buckling strut response. The ISO equation incorporates several factors that you can define. Effective and added length factors account for element end fixity, and buckling reduction factors account for bending moment influence on buckling. You can define nondefault values for these factors in each local cross-section direction.

Input File Usage: To allow switching to buckling strut response with default coefficients for the ISO equation and the default buckling envelope, use the following option:

**FRAME SECTION*, SECTION=PIPE, ELSET=*name*, BUCKLING,
YIELD STRESS=σ^0

To allow switching to buckling strut response with nondefault coefficients for the ISO equation and the default buckling envelope, use all of the following options:

**FRAME SECTION*, SECTION=PIPE, ELSET=*name*, BUCKLING,
YIELD STRESS=σ^0
**BUCKLING LENGTH*
**BUCKLING REDUCTION FACTORS*

To allow switching to buckling strut response with nondefault coefficients for the ISO equation and a user-defined buckling envelope, use all of the following options:

**FRAME SECTION*, SECTION=PIPE, ELSET=*name*, YIELD STRESS=σ^0
**BUCKLING ENVELOPE*
**BUCKLING LENGTH*
**BUCKLING REDUCTION FACTORS*

Defining the Reference Temperature for Thermal Expansion

You can define a thermal expansion coefficient for the frame section. The thermal expansion coefficient may be temperature dependent. In this case you must define the reference temperature for thermal expansion, θ^0.

Input File Usage: Use both of the following options:

**FRAME SECTION*, ZERO=θ^0
**THERMAL EXPANSION*

Specifying Temperature and Field Variables

Define temperatures and field variables by giving the value at the origin of the cross-section (i.e., only one temperature or field-variable value is given).

Input File Usage: Use one or more of the following options:

**TEMPERATURE*
**FIELD*
**INITIAL CONDITIONS*, TYPE=TEMPERATURE
**INITIAL CONDITIONS*, TYPE=FIELD

Frame Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the frame elements available in Abaqus/Standard.

References:

- *Frame Elements*
- **FRAME SECTION*

Element Types

Frame in a Plane

FRAME2D 2-node straight frame element

Active Degrees of Freedom

1, 2, 6

Additional Solution Variables

Two additional variables relating to the axial and lateral displacements.

Frame in Space

FRAME3D 2-node straight frame element

Active Degrees of Freedom

1, 2, 3, 4, 5, 6

Additional Solution Variables

Three additional variables relating to the axial and lateral displacements.

Nodal Coordinates Required

Frame in a plane: *X*, *Y* (Direction cosines of the normal are not used; any values given are ignored.)

Frame in space: *X*, *Y*, *Z* (Direction cosines of the normal are not used; any values given are ignored.)

Element Property Definition

Local orientations defined as described in *Orientations* cannot be used with frame elements to define local material directions. The orientation of the local section axes in space is discussed in *Frame Elements*.

Input File Usage:

**FRAME SECTION*

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): GRAV
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): PX
Units: FL^{-1}
Description: Force per unit length in global *X*-direction.

Load ID (*DLOAD): PY
Units: FL^{-1}
Description: Force per unit length in global *Y*-direction.

Load ID (*DLOAD): PZ
Units: FL^{-1}
Description: Force per unit length in global *Z*-direction (only for frames in space).

Load ID (*DLOAD): P1
Units: FL^{-1}
Description: Force per unit length in frame local *1*-direction (only for frames in space).

Load ID (*DLOAD): P2
Units: FL^{-1}
Description: Force per unit length in frame local *2*-direction.

Abaqus/Aqua Loads

Abaqus/Aqua loads are specified as described in *Abaqus/Aqua Analysis*.

Load ID (*CLOAD/ *DLOAD): FDD
Units: FL^{-1}
Description: Transverse fluid drag load.

Load ID (*CLOAD/ *DLOAD): FD1
Units: F
Description: Fluid drag force on the first end of the frame (node 1).

Load ID (*CLOAD/ *DLOAD): FD2

Units: F
Description: Fluid drag force on the second end of the frame (node 2).

Load ID (*CLOAD/ *DLOAD): FDT
Units: FL^{-1}
Description: Tangential fluid drag load.

Load ID (*CLOAD/ *DLOAD): FI
Units: FL^{-1}
Description: Transverse fluid inertia load.

Load ID (*CLOAD/ *DLOAD): FI1
Units: F
Description: Fluid inertia force on the first end of the frame (node 1).

Load ID (*CLOAD/ *DLOAD): FI2
Units: F
Description: Fluid inertia force on the second end of the frame (node 2).

Load ID (*CLOAD/ *DLOAD): PB
Units: FL^{-1}
Description: Buoyancy load (closed-end condition).

Load ID (*CLOAD/ *DLOAD): WDD
Units: FL^{-1}
Description: Transverse wind drag load.

Load ID (*CLOAD/ *DLOAD): WD1
Units: F
Description: Wind drag force on the first end of the frame (node 1).

Load ID (*CLOAD/ *DLOAD): WD2
Units: F
Description: Wind drag force on the second end of the frame (node 2).

Foundations

Foundations are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): FX
Units: FL^{-2}
Description: Stiffness per unit length in global *X*-direction.

Load ID (*FOUNDATION): FY
Units: FL^{-2}
Description: Stiffness per unit length in global *Y*-direction.

Load ID (*FOUNDATION): FZ
Units: FL^{-2}
Description: Stiffness per unit length in global *Z*-direction (only for frames in space).

Load ID (*FOUNDATION): F1
Units: FL^{-2}
Description: Stiffness per unit length in frame local *1*-direction (only for frames in space).

Load ID (*FOUNDATION): F2
Units: FL^{-2}
Description: Stiffness per unit length in frame local *2*-direction.

Element Output

All element output variables are given at the element ends (nodes 1 and 2) and midpoint (node 3).

Section Forces and Moments

SF1 Axial force.

SF2 Transverse shear force in the local 2-direction.

SF3 Transverse shear force in the local 1-direction (only available for frames in space).

SM1 Bending moment about the local 1-axis.

SM2 Bending moment about the local 2-axis (only available for frames in space).

SM3 Twisting moment about the frame axis (only available for frames in space).

See *Frame elements with lumped plasticity* for a discussion of the section forces and moments.

Section Elastic Strains and Curvatures

SEE1 Elastic axial strain.

SKE1 Elastic curvature change about the local 1-axis.

SKE2 Elastic curvature change about the local 2-axis (only available for frames in space).

SKE3 Elastic twist of the beam (only available for frames in space).

Plastic Displacements and Rotations in the Element Coordinate System

SEP1 Plastic axial displacement.

SKP1 Plastic rotation about the local 1-axis.

SKP2 Plastic rotation about the local 2-axis (only available for frames in space).

SKP3 Plastic rotation about the beam axis (only available for frames in space).

Section Force and Moment Backstresses

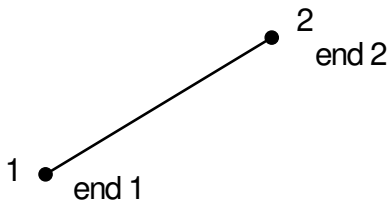
SALPHA1 Axial force backstress.

SALPHA2 Bending moment backstress about the local 1-axis.

SALPHA3 Bending moment backstress about the local 2-axis (only available for frames in space).

SALPHA4 Twisting moment backstress about the beam axis (only available for frames in space).

Node Ordering on Elements



2 - node element

For frames in space an additional node may be given after a frame element's connectivity (in the element definition—see *Element Definition*) to define the approximate direction of the first cross-section axis, $\mathbf{n}_1$. See *Frame Elements* for details.

Elbow Elements

In this section:

- *Pipes and Pipebends with Deforming Cross-Sections: Elbow Elements*
- *Elbow Element Library*

Pipes and Pipebends with Deforming Cross-Sections: Elbow Elements

Products: Abaqus/Standard

References:

- *Elbow Element Library*
- **BEAM SECTION*

Overview

Elbow elements:

- are intended to provide accurate modeling of the nonlinear response of initially circular pipes and pipebends when distortion of the cross-section by ovalization and warping dominates the behavior;
- appear as beams but are shells with quite complex deformation patterns allowed;
- use plane stress theory to model the deformation through the pipe wall; and
- cannot provide nodal values of stress, strain, and other constitutive results.

Typical Applications

In the usual approach to linear analysis of elbows, the response prediction is based on semianalytical results, used as "flexibility factors" to correct results obtained with simple beam theory. Such factors do not apply in nonlinear cases, and the pipeline must be modeled as a shell to predict the response accurately (for example, see *Parametric study of a linear elastic pipeline under in-plane bending*). Although the elbow elements appear as beams, they are, in fact, shells, with quite complex deformation patterns allowed. In thin-walled elbows the interaction of elbows and adjacent straight segments is an important aspect of elbow modeling, as are the large rotations that readily occur in the cross-sectional deformation, even at small relative rotations of the pipe axis itself. All of these effects (including the stiffening effect of internal pressure) can be modeled with these elements.

Elbow elements are intended to provide accurate modeling of the nonlinear response of initially circular pipes and pipebends when distortion of the cross-section by ovalization and warping dominates the behavior. Such behavior arises in two circumstances: in pipebends, where the initial curvature of the pipe, together with the thinness of the wall of the pipe, causes ovalization to dominate the response, and in straight pipe sections, where excessive bending can lead to a buckling collapse of the thin-walled circular section ("Brazier buckling").

Because the elbow elements use a full shell formulation around the circumference, the number of degrees of freedom per element is high. Elbow elements that use all Fourier modes (discussed below) to model ovalization and warping are considerably more expensive computationally than beam elements, but their cost is comparable to that of coarse shell models, which can be used to model the section.

If an analysis requires connecting pipe elements to a pipebend, it is easier to connect elbow elements to pipe elements than it is to connect shell elements to pipe elements.

Choosing an Appropriate Element

Elbow elements use polynomial interpolation along their length (linear or quadratic depending on the element type), together with Fourier interpolation around the pipe to model the ovalization and warping of the section. Shell theory is then used to model the behavior.

Two types of elbow elements are provided.

ELBOW31 and ELBOW32

Element types ELBOW31 and ELBOW32 are the most complete elbow elements. In these elements the ovalization of the pipe wall is made continuous from one element to the next, thus modeling such effects as the interaction between pipe bends (elbows) and adjacent straight segments of the pipeline.

ELBOW31 and ELBOW32 should not be used for the analysis of unconnected straight pipes unless the warping and ovalization are restrained at some point in the pipe.

ELBOW31B and ELBOW31C

Element types ELBOW31B and ELBOW31C use a simplified version of the formulation, in which ovalization only is considered (no warping) and axial gradients of the ovalization are neglected. These approximations are often satisfactory, and indeed they form the basis of the standard flexibility factor approach used in linear analysis of piping systems. They provide a considerably less expensive capability. ELBOW31C includes the further approximation that the odd numbered terms in the Fourier interpolation around the pipe, except the first term, are neglected. This formulation provides a slightly less expensive model for cases where the radius of the pipe is small compared to the radius of curvature of the pipe axis.

Defining the Elements Section Properties

You use a beam section definition integrated during the analysis to define the section properties of elbow elements. Give the outside radius of the pipe, r; pipe wall thickness, t; and elbow torus radius, measured to the pipe axis, R. For a straight pipe, set R to zero.

You must associate these properties with a set of elbow elements.

Input File Usage: **BEAM SECTION*, SECTION=ELBOW, ELSET=*name*
r, *t*, *R*

Defining the Section Orientation

For all elbow elements the section must be oriented in space by specifying a point that, together with the nodes of the element, defines the plane of the $\mathbf{a}_2$-axis in *Figure 1*. For bent pipes this point should lie outside the bend (the side of the pipe on the outside of the bend is referred to as the *extrados*). For pipebends of less than 180° this point can be set to be the point of intersection of the tangents to the adjacent straight pipe runs. If a pipebend subtends an angle greater than or equal to 180°, the bend should be partitioned into sections of less than 180° and a separate beam section should be defined for each partition so that the point used to define the plane of the $\mathbf{a}_2$-axis can lie outside of the extrados. When the elements are used to model straight pipes, the point can be any point off the pipe axis.

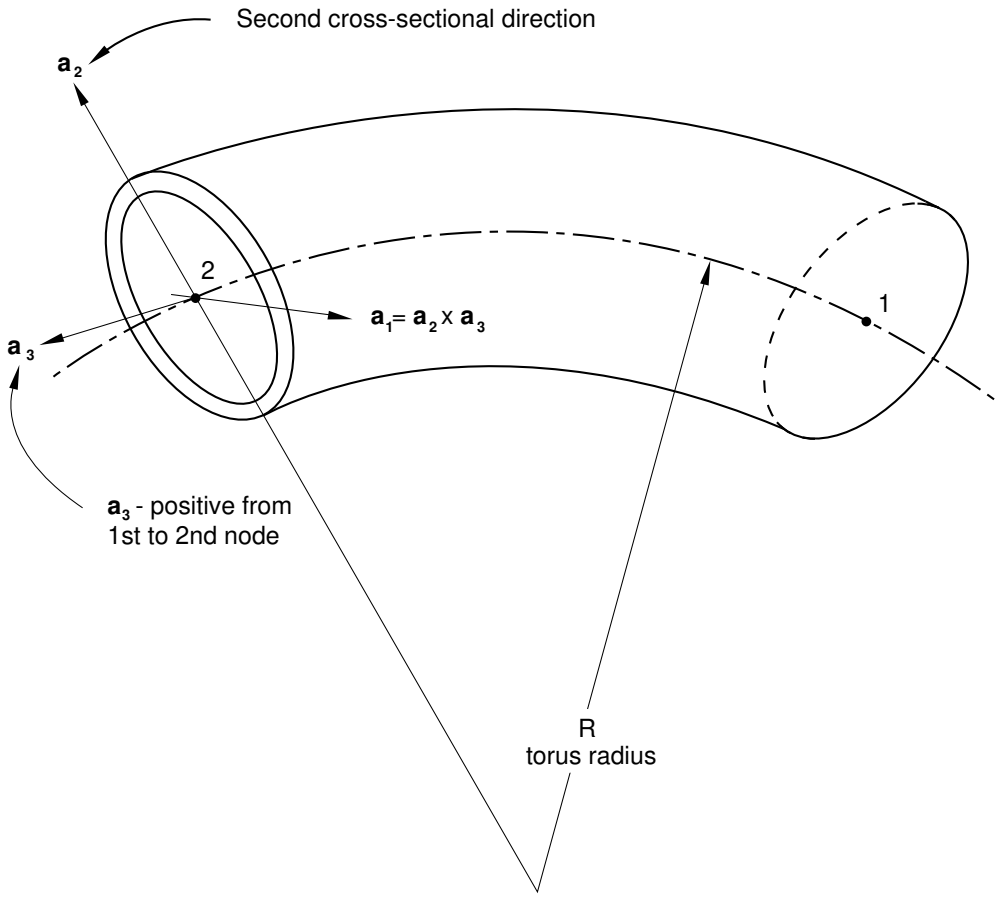


Figure 1: Elbow element geometry.

When ovalization modeling is extended onto straight runs adjacent to a pipebend by using ELBOW31 or ELBOW32 elements for the pipebend and for the straight pipe, you must ensure that the $\mathbf{a_2}$-axis is defined so that its orientation about the axis of the pipe is the same between the pipebend and each of the straight segments. When possible, the $\mathbf{a_2}$-axis should also be the same between adjacent pipebends. In some cases, such as adjacent pipebends in different planes, the $\mathbf{a_2}$-axes are necessarily discontinuous. In such cases separate nodes must be introduced at the point where the $\mathbf{a_2}$-axis changes orientation, and MPC type ELBOW must be invoked to impose the appropriate constraints to ensure continuity of displacements. See *Figure 2*.

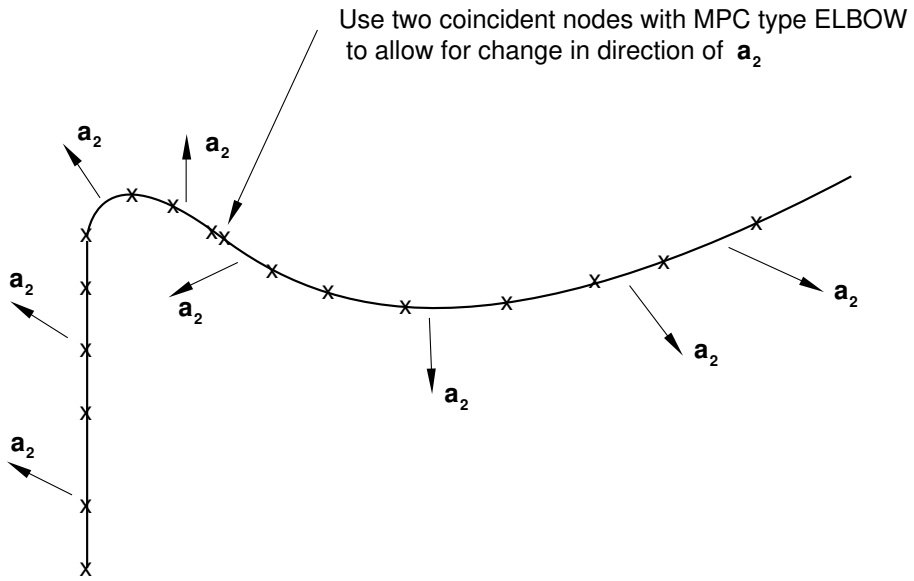


Figure 2: Use of MPC type ELBOW with ELBOW31 or ELBOW32.

Input File Usage: **BEAM SECTION*, SECTION=ELBOW
first data line
coordinates of orientation point

Defining the Number of Integration Points and Fourier Modes

You can specify the number of integration points and Fourier modes for an elbow section. Experience suggests that for relatively thick-walled cases 4 Fourier modes with 12 integration points around the pipe are sufficient. For thin-walled elbows 6 Fourier modes and 18 integration points around the pipe are needed. As a general rule, the number of integration points around the pipe should not be less than three times the number of Fourier modes used; otherwise, singularities may arise in the stiffness matrix. When used with zero Fourier modes, the elements become simple pipe elements with hoop strain and stress included: when Poisson's ratio is set to zero, they show similar behavior to the PIPE elements in Abaqus (see *Choosing a Beam Element*).

Input File Usage: **BEAM SECTION*, SECTION=ELBOW
first data line
second data line
number of int. pts. through thickness, *number of int. pts. around pipe*, *number of Fourier modes*

Assigning a Material Definition to a Set of Elbow Elements

You must associate a material definition with each elbow section definition.

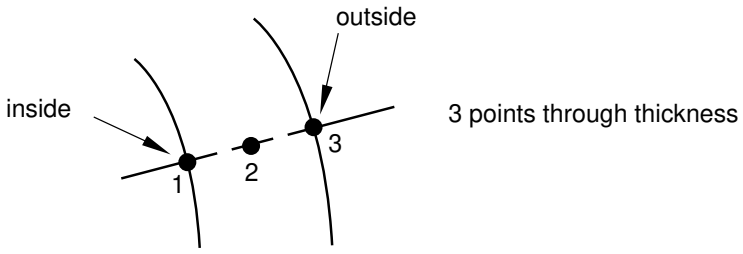
Input File Usage: **BEAM SECTION*, SECTION=ELBOW, MATERIAL=*name*

Specifying Temperature and Field Variables

Temperature and field variables can be specified by defining the values at specific points through the section or by defining the value at the middle of the pipe wall and specifying the gradient through the pipe thickness.

By Defining the Values at Points through the Section

You can define temperatures and field variables by giving the values at each of the three points shown below.



No matter how many section points there are through the thickness of the elbow, specify the values at only these three points. These three values are applied to all integration points around the circumference so that the only admissible variation is in the radial direction.

Input File Usage: **BEAM SECTION*, SECTION=ELBOW, TEMPERATURE=VALUES

By Defining the Value at the Middle of the Pipe Wall and the Gradient through the Thickness

Alternatively, you can define temperatures and field variables by giving the value on the middle surface of the pipe wall and the gradient of temperature with respect to position through the pipe wall thickness, positive when the outside surface is hotter than the inside surface.

Input File Usage: **BEAM SECTION*, SECTION=ELBOW, TEMPERATURE=GRADIENTS

Using Elbow Elements in Large-Displacement Analysis

When elbow elements are subjected to pipe pressure loads (load types PI, PE, HPI, or HPE) in large-displacement analysis (*General and Perturbation Procedures*), the most significant contributions to the load stiffness are taken into account.

Defining Kinematic Boundary Conditions on Elbow Elements

Kinematic boundary conditions on the standard degrees of freedom at the nodes of elbow elements (that is, degrees of freedom 1–6) should be treated in the usual way.

In addition, the elements have ovalization and warping terms stored internally. For ELBOW31B and ELBOW31C elements this requires no additional consideration. For ELBOW31 or ELBOW32 elements you may often need to provide kinematic boundary conditions on these additional degrees of freedom. For example, it is common to model a pipeline with ovalization and warping allowed in the elbows and adjacent straight pipe segments but no ovalization in the middle segments of long, straight pipe runs (see *Figure 3*).

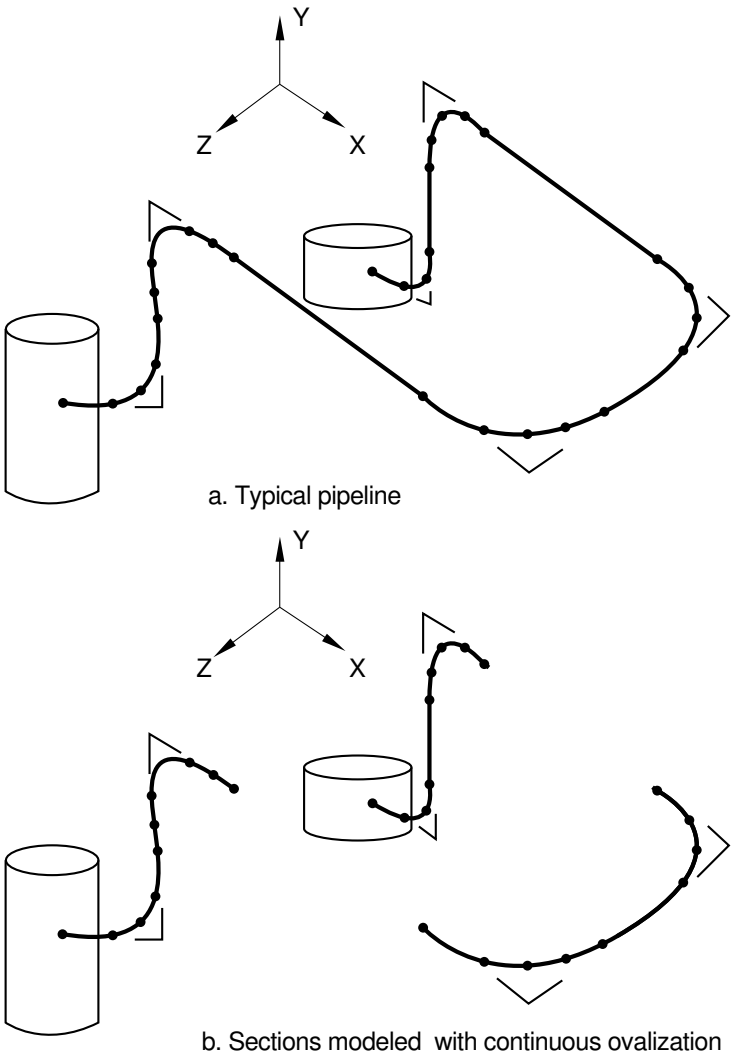


Figure 3: Pipeline schematic.

(The latter is usually accomplished by specifying ELBOW31 elements with zero modes or PIPE31 elements so that the usual bending terms and the uniform radial expansion term, associated with pressure in the pipe, are included; if internal pressure is not important, a simple beam element, B31, can be used instead.) Where the segments with ovalization and warping end, the warping must be restrained; and if a stiff flange or vessel exists at that point, the ovalization should also be restrained. To do so, specify NOWARP and/or NOOVAL or NODEFORM boundary conditions for that node (*Boundary Conditions*).

NOWARP means that no warping is allowed at the node, but ovalization and uniform radial expansion are allowed; NOOVAL means that there can be no ovalization at the node, but warping and uniform radial expansion are allowed; NODEFORM means that there can be no cross-section deformation at all—no warping, ovalization, or uniform radial expansion.

Typically, NOWARP will be specified at the end of a pipebend segment modeled with ELBOW31 adjacent to a straight pipe run, while NOWARP and NOOVAL would be specified at a stiff flange or vessel attachment point. NODEFORM restrains all cross-sectional deformation, including the uniform radial expansion term: this will result in large stresses if thermal expansion occurs. NODEFORM should be used, for example, at a built-in end.

Visualizing the Cross-Section Deformation

The current release of Abaqus/Standard does not provide a direct way of visualizing the cross-section ovalization. However, the utility routine `felbow.f` (*Creation of a data file to facilitate the postprocessing of elbow element results: FELBOW*) creates a data file that can be used in Abaqus/CAE to plot the current coordinates of the integration points around the circumference of the elbow section of interest. The routine uses output variable COORD (*Abaqus/Standard Output Variable Identifiers*) to obtain the current coordinates of the integration points. These values are available only if geometric nonlinearity is considered in the step. You will have to ensure that the variable COORD is written to the results (`.fil`) file for this purpose.

The routine is suitable for elbow elements oriented arbitrarily in space: the integration points of the elbow section are projected appropriately to a coordinate system suitable for plotting the cross-section. The input data for plotting are written to a file that can be read into Abaqus/CAE. An *X–Y* plot of the elbow element's deformed cross-section can be displayed using the **XY Data Manager** in the Visualization module.

In addition to facilitating the visualization of the cross-section ovalization, the program also allows you to create data files to plot the variation of a variable along a line of elbow elements and around the circumference of a given elbow element.

Similar C++ and Python utility routines, `felbow.C` (*A C++ version of FELBOW*) and `felbow.py` (*An Abaqus Scripting Interface version of FELBOW*), are provided to process the pertinent elbow element results output written to the output database (`.odb`) file. When these programs are executed, they write data to an ASCII format file and/or an output database file that can be used in Abaqus/CAE to plot the current coordinates of the integration points around the circumference of the elbow section. Both these routines can also be used to visualize the variation of an output variable around the circumference of the elbow section.

Elbow Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the elbow elements available in Abaqus/Standard.

References:

- *Pipes and Pipebends with Deforming Cross-Sections: Elbow Elements*
- **BEAM SECTION*

Element Types

ELBOW31	2-node pipe in space with deforming section, linear interpolation along the pipe
ELBOW32	3-node pipe in space with deforming section, quadratic interpolation along the pipe
ELBOW31B	2-node pipe in space with ovalization only, axial gradients of ovalization neglected
ELBOW31C	2-node pipe in space with ovalization only, axial gradients of ovalization neglected. This formulation is the same as that for element type ELBOW31B, with the exception that all odd numbered terms in the Fourier interpolation around the pipe but the first term are neglected.

Active Degrees of Freedom

1, 2, 3, 4, 5, 6

Additional Solution Variables

Elbow elements have numerous variables to model cross-sectional ovalization and warping. The number of variables depends on the type of elbow element, the number of nodes, and the number of Fourier modes chosen. In the following table p is the number of Fourier modes:

Element type	Number of variables
ELBOW31	16, if p=0
	(16p+8), if $p \geq 1$
ELBOW32	24, if p=0
	(24p+12), if $p \geq 1$
ELBOW31B	13+2p, if p=0,1
	11+4p, if $p \geq 2$
ELBOW31C	13+2p, if p=0,1,3,5

Element type	Number of variables
	15+2p, if p=2,4,6

Nodal Coordinates Required

X, Y, Z

Element Property Definition

Input File Usage: **BEAM SECTION*, SECTION=ELBOW

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Units: FL^{-3}
Description: Body force per unit volume in global *X*-direction.

Load ID (*DLOAD): BY
Units: FL^{-3}
Description: Body force per unit volume in global *Y*-direction.

Load ID (*DLOAD): BZ
Units: FL^{-3}
Description: Body force per unit volume in global *Z*-direction.

Load ID (*DLOAD): BXNU
Units: FL^{-3}
Description: Nonuniform body force in global *X*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BYNU
Units: FL^{-3}
Description: Nonuniform body force in global *Y*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BZNU
Units: FL^{-3}
Description: Nonuniform body force in global *Z*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): CENT
Units: $FL^{-4}(ML^{-3}T^{-2})$

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume and ω is the angular velocity).

Load ID (*DLOAD): CENTRIF

Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): GRAV
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): HPE
Units: FL^{-2}
Description: Hydrostatic external pressure, with linear variation in global *Z* (closed-end condition).

Load ID (*DLOAD): HPI
Units: FL^{-2}
Description: Hydrostatic internal pressure, with linear variation in global *Z* (closed-end condition).

Load ID (*DLOAD): PE
Units: FL^{-2}
Description: Uniform external pressure (closed-end condition).

Load ID (*DLOAD): PI
Units: FL^{-2}
Description: Uniform internal pressure (closed-end condition).

Load ID (*DLOAD): PENU
Units: FL^{-2}
Description: Nonuniform external pressure with magnitude supplied via user subroutine *DLOAD* (closed-end condition).

Load ID (*DLOAD): PINU
Units: FL^{-2}
Description: Nonuniform internal pressure with magnitude supplied via user subroutine *DLOAD* (closed-end condition).

Load ID (*DLOAD): ROTA
Units: T^{-2}
Description: Rotary acceleration load (magnitude is input as α, where α is the rotary acceleration).

Abaqus/Aqua Loads

Abaqus/Aqua loads are specified as described in *Abaqus/Aqua Analysis*.

Load ID (*CLOAD/ *DLOAD): FDD
Units: FL^{-1}
Description: Transverse fluid drag load.

Load ID (*CLOAD/ *DLOAD): FD1
Units: F
Description: Fluid drag force on the first end of the elbow (node 1).

Load ID (*CLOAD/ *DLOAD): FD2
Units: F
Description: Fluid drag force on the second end of the elbow (node 2 or node 3).

Load ID (*CLOAD/ *DLOAD): FDT
Units: FL^{-1}

Description: Tangential fluid drag load.

Load ID (*CLOAD/ *DLOAD): FI
Units: FL^{-1}
Description: Transverse fluid inertia load.

Load ID (*CLOAD/ *DLOAD): FI1
Units: F
Description: Fluid inertia force on the first end of the elbow (node 1).

Load ID (*CLOAD/ *DLOAD): FI2
Units: F
Description: Fluid inertia force on the second end of the elbow (node 2 or node 3).

Load ID (*CLOAD/ *DLOAD): PB
Units: FL^{-1}
Description: Buoyancy force (closed-end condition).

Load ID (*CLOAD/ *DLOAD): WDD
Units: FL^{-1}
Description: Transverse wind drag load.

Load ID (*CLOAD/ *DLOAD): WD1
Units: F
Description: Wind drag force on the first end of the elbow (node 1).

Load ID (*CLOAD/ *DLOAD): WD2
Units: F
Description: Wind drag force on the second end of the elbow (node 2 or node 3).

Element Output

The default stress output points are on the inside surface and the outside surface at all integration stations around the pipe.

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11 Direct stress along the pipe.

S22 Direct stress around the pipe section.

S12 Shear stress in the pipe wall.

Section Forces and Moments

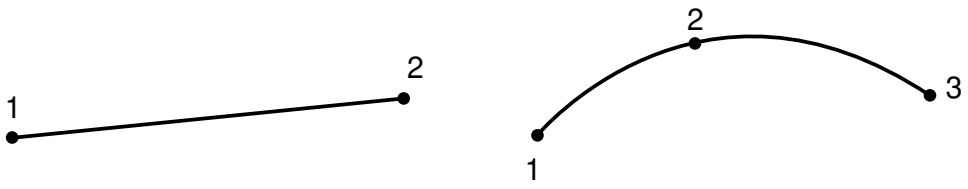
SF1 Axial force.

SM1 Bending moment about the local 1-axis.

SM2 Bending moment about the local 2-axis.

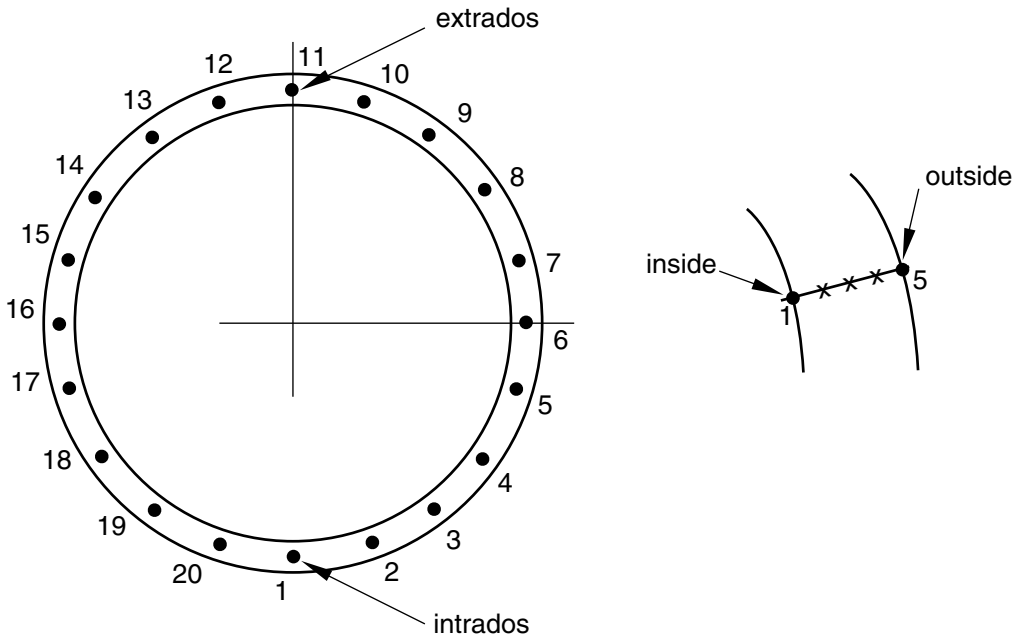
SM3 Twisting moment about the elbow axis.

Node Ordering on Elements



2-node element 3-node element

Numbering of Integration Points for Output



The extrados is the side of the pipebend that is furthest away from the center of the torus defining the pipebend; that is, the side of the pipebend to which the $\mathbf{a}_2$-axis points. The intrados is the side of the pipebend closest to the center of the torus.

The middle surface integration points around a section are shown above. There is a default of five thickness direction integration points at each such point, with point 1 on the inside surface of the pipe and point 5 on the outside surface.

For ELBOW31 and ELBOW31B only one integration station is used along the axis of the element. For ELBOW32 two integration stations are used along the axis of the elbow and the point numbers on the second section are a continuation of those on the first section (e.g., 21, 22, …, 40 in the default case), located around the pipe as shown above.

Shell Elements

In this section:

- *About Shell Elements*
- *Choosing a Shell Element*
- *Defining the Initial Geometry of Conventional Shell Elements*
- *Shell Section Behavior*
- *Using a Shell Section Integrated during the Analysis to Define the Section Behavior*
- *Using a General Shell Section to Define the Section Behavior*
- *Three-Dimensional Conventional Shell Element Library*
- *Continuum Shell Element Library*
- *Axisymmetric Shell Element Library*
- *Axisymmetric Shell Elements with Nonlinear, Asymmetric Deformation*

About Shell Elements

Overview

Shell modeling consists of:

- choosing the appropriate shell element type (*Choosing a Shell Element*);
- defining the initial geometry of the surface (*Defining the Initial Geometry of Conventional Shell Elements*);
- determining whether or not numerical integration is needed to define the shell section behavior (*Shell Section Behavior*); and
- defining the shell section behavior (*Using a Shell Section Integrated during the Analysis to Define the Section Behavior* or *Using a General Shell Section to Define the Section Behavior*).

Abaqus offers a wide variety of shell modeling options.

References:

Conventional Shell Versus Continuum Shell

Shell elements are used to model structures in which one dimension, the thickness, is significantly smaller than the other dimensions. Conventional shell elements use this condition to discretize a body by defining the geometry at a reference surface. In this case the thickness is defined through the section property definition. Conventional shell elements have displacement and rotational degrees of freedom.

In contrast, continuum shell elements discretize an entire three-dimensional body. The thickness is determined from the element nodal geometry. Continuum shell elements have only displacement degrees of freedom. From a modeling point of view continuum shell elements look like three-dimensional continuum solids, but their kinematic and constitutive behavior is similar to conventional shell elements.

Figure 1 illustrates the differences between a conventional shell and a continuum shell element.

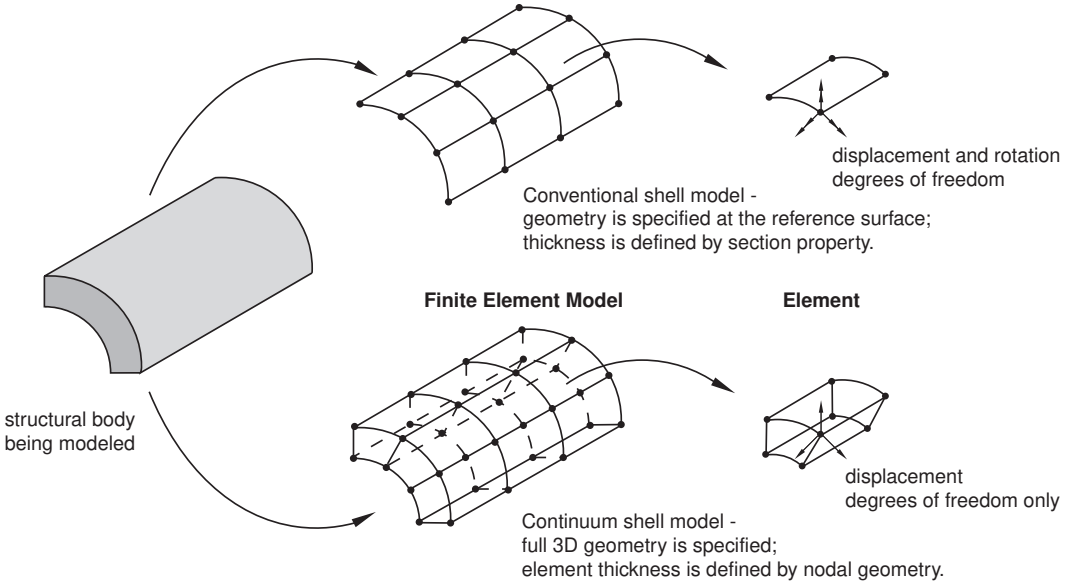


Figure 1: Conventional versus continuum shell element.

Conventions

The conventions that are used for shell elements are defined below.

Definition of Local Directions on the Surface of a Shell in Space

The default local directions used on the surface of a shell for definition of anisotropic material properties and for reporting stress and strain components are defined in *Conventions*. You can define other directions by defining a local orientation (see *Orientations*), except for SAX1, SAX2, and SAX2T elements (*Axisymmetric Shell Element Library*), which do not support orientations. A spatially varying local coordinate system defined with a distribution (*Distribution Definition*) can be assigned to shell elements. For SAXA elements (*Axisymmetric Shell Elements with Nonlinear, Asymmetric Deformation*) any anisotropic material definition must be symmetric with respect to the r–z plane at $\theta = 0$ and π.

In large-deformation (geometrically nonlinear) analysis these local directions rotate with the average rotation of the surface at that point. They are output as directions in the current configuration except in the shell elements in Abaqus/Standard that provide only large rotation but small strain (element types STRI3, STRI65, S4R5, S8R, S8RT, S8R5, S9R5—see *Choosing a Shell Element*), where they are output as directions in the reference configuration. Therefore, in geometrically nonlinear analysis, when displaying these directions or when displaying principal values of stress, strain, or section forces or moments in Abaqus/CAE, the current (deformed) configuration should be used except for the small-strain elements in Abaqus/Standard, for which the reference configuration should be used.

Positive Normal Definition for Conventional Shell Elements

The “top” surface of a conventional shell element is the surface in the positive normal direction and is referred to as the positive (SPOS) face for contact definition. The “bottom” surface is in the negative direction along the normal and is referred to as the negative (SNEG) face for contact definition. Positive and negative are also used to designate top and bottom surfaces when specifying offsets of the reference surface from the shell's midsurface.

The positive normal direction defines the convention for pressure load application and output of quantities that vary through the thickness of the shell. A positive pressure load applied to a shell element produces a load that acts in the direction of the positive normal.

Three-Dimensional Conventional Shells

For shells in space the positive normal is given by the right-hand rule going around the nodes of the element in the order that they are specified in the element definition. See *Figure 2*.

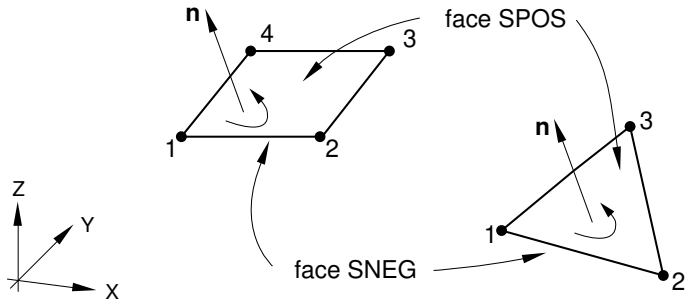


Figure 2: Positive normals for three-dimensional conventional shells.

Axisymmetric Conventional Shells

For axisymmetric conventional shells (including the SAXA1*n* and SAXA2*n* elements that allow for nonsymmetric deformation) the positive normal direction is defined by a 90° counterclockwise rotation from the direction going from node 1 to node 2. See *Figure 3*.

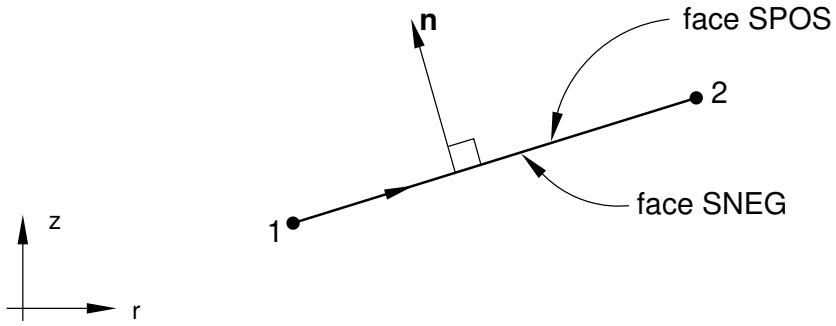


Figure 3: Positive normal for conventional axisymmetric shells.

Normal Definition for Continuum Shell Elements

Figure 4 illustrates the key geometrical features of continuum shells.

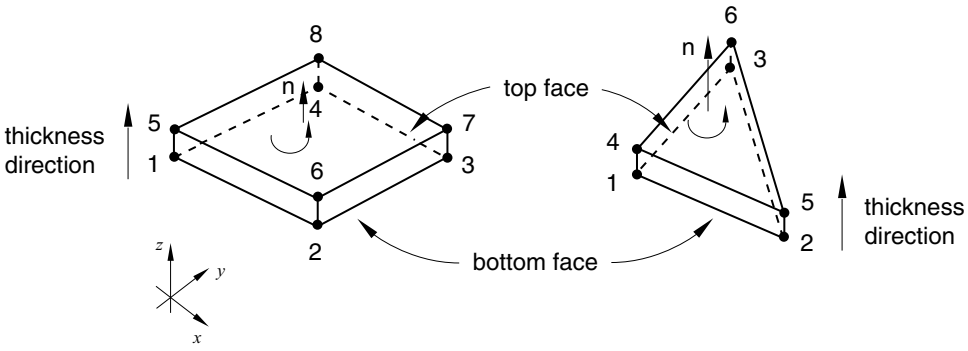


Figure 4: Default normals and thickness direction for continuum shell elements.

It is important that the continuum shells are oriented properly, since the behavior in the thickness direction is different from that in the in-plane directions. By default, the element top and bottom faces and, hence, the element

normal, stacking direction, and thickness direction are defined by the nodal connectivity. For the triangular in-plane continuum shell element (SC6R) the face with corner nodes 1, 2, and 3 is the bottom face; and the face with corner nodes 4, 5, and 6 is the top face. For the quadrilateral continuum shell element (SC8R) the face with corner nodes 1, 2, 3, and 4 is the bottom face; and the face with corner nodes 5, 6, 7, and 8 is the top face. The stacking direction and thickness direction are both defined to be the direction from the bottom face to the top face. Additional options for defining the element thickness direction, including one option that is independent of nodal connectivity, are presented below.

Surfaces on continuum shells can be defined by specifying the face identifiers S1–S6 identifying the individual faces as defined in *Continuum Shell Element Library*. Free surface generation can also be used.

Pressure loads applied to faces P1–P6 are defined similar to continuum elements, with a positive pressure directed into the element.

Defining the Stacking and Thickness Direction

By default, the continuum shell stacking direction and thickness direction are defined by the nodal connectivity as illustrated in *Figure 4*. Alternatively, you can define the element stacking direction and thickness direction by either selecting one of the element's isoparametric directions or by using an orientation definition.

Defining the Stacking and Thickness Direction Based on the Element Isoparametric Direction

You can define the element stacking direction to be along one of the element's isoparametric directions (see *Figure 5* for element stack directions). The 8-node hexahedron continuum shell has three possible stacking directions; the 6-node in-plane triangular continuum shell has only one stack direction, which is in the element 3-isoparametric direction. The default stacking direction is 3, providing the same thickness and stacking direction as outlined in the previous section.

To obtain a desired thickness direction, the choice of the isoparametric direction depends on the element connectivity. For a mesh-independent specification, use an orientation-based method as described below.

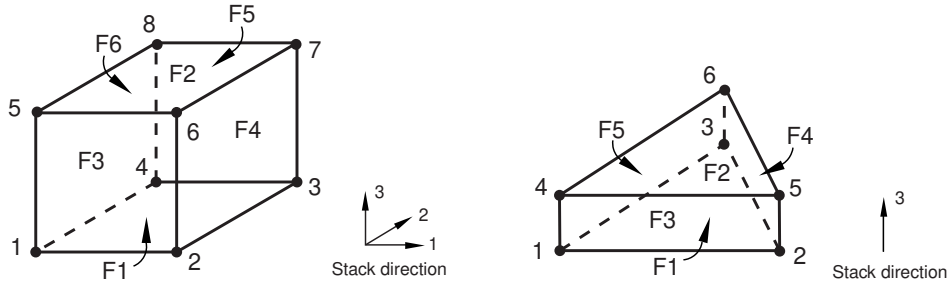


Figure 5: Stack directions for SC6R and SC8R elements.

Input File Usage: Use one of the following options to define the element stacking direction based on the element's isoparametric directions:

**SHELL SECTION*, STACK DIRECTION=*n*
**SHELL GENERAL SECTION*, STACK DIRECTION=*n*

where *n* = 1, 2, or 3.

Abaqus/CAE Usage: Use the following option to define the stacking direction based on the element's isoparametric directions if the continuum shell is defined using a composite layup:

Property module: **Create Composite Layup**: select **Continuum Shell** as the **Element Type**: **Stacking Direction**: **Element direction 1**, **Element direction 2**, or **Element direction 3**

Use the following option to define the stacking direction based on the element's isoparametric directions if the continuum shell is defined using a composite shell section:

Assign->Material Orientation: select regions: **Use Default Orientation or Other Method**: **Stacking Direction**: **Element isoparametric direction 1**, **Element isoparametric direction 2**, or **Element isoparametric direction 3**

Defining the Stacking and Thickness Direction Based on an Orientation Definition

Alternatively, you can define the element stacking direction based on a local orientation definition. For shell elements the orientation definition defines an axis about which the local 1 and 2 material directions may be rotated. This axis also defines an approximate normal direction. The element stacking and thickness directions are defined to be the element isoparametric direction that is closest to this approximate normal (see *Figure 6*).

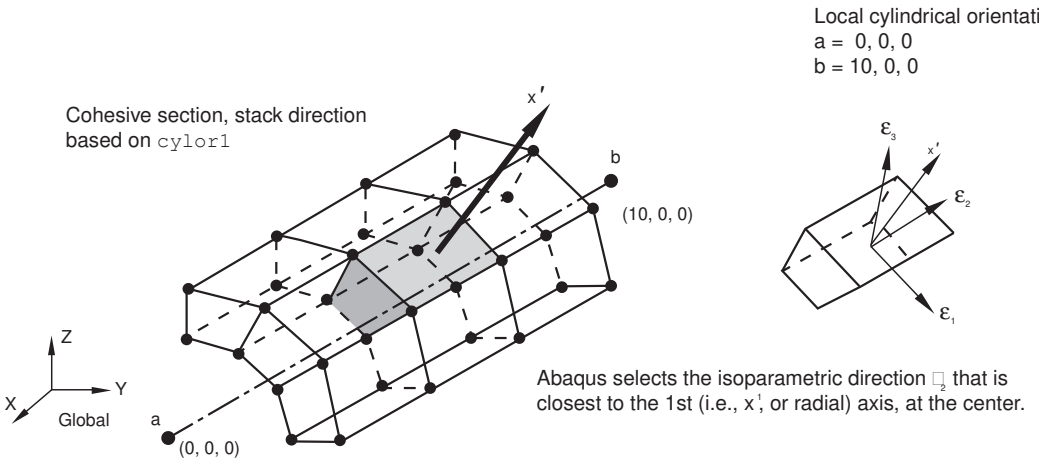


Figure 6: Example illustrating the use of a cylindrical system to define the stacking direction.

The pinched cylinder problem and *LE3: Hemispherical shell with point loads* illustrate the use of a cylindrical and spherical orientation system, respectively, to define the stack and thickness direction independent of nodal connectivity.

Input File Usage: Use one of the following options to define the element stacking direction based on a user-defined orientation:

**SHELL SECTION*, STACK DIRECTION=ORIENTATION, ORIENTATION=*name*
**SHELL GENERAL SECTION*, STACK DIRECTION=ORIENTATION, ORIENTATION=*name*

Abaqus/CAE Usage: Use the following option to define the stacking direction based on a user-defined orientation if the continuum shell is defined using a composite layup:

Property module: **Create Composite Layup**: select **Continuum Shell** as the **Element Type**: **Stacking Direction**: **Layup orientation**

Use the following option to define the stacking direction based on a user-defined orientation if the continuum shell is defined using a composite shell section:

Assign->Material Orientation: select regions: **Use Default Orientation or Other Method**: **Stacking Direction**: **Normal direction of material orientation**

Verifying the Element Stack and Thickness Direction

You can verify the element stack and thickness direction visually in Abaqus/CAE by either contouring the element section thickness or plotting the material axis. Generally, the in-plane dimensions are significantly larger than the element thickness. By contouring the shell section thickness, output variable STH, you can easily verify that all elements are oriented appropriately and have the correct thickness. If the element is oriented improperly, one of the in-plane dimensions will become the element section thickness, resulting in a discontinuous contour plot.

Alternatively, you can plot the material axis to verify that the 3-axis points in the desired normal direction. If the element is oriented improperly, one of the in-plane axes (either the 1- or 2-axis) would point in the normal direction.

Numbering of Section Points through the Shell Thickness

The section points through the thickness of the shell are numbered consecutively, starting with point 1. For shell sections integrated during the analysis, section point 1 is exactly on the bottom surface of the shell if Simpson's rule is used, and it is the point that is closest to the bottom surface if Gauss quadrature is used. For general shell sections, section point 1 is always on the bottom surface of the shell.

For a homogeneous section the total number of section points is defined by the number of integration points through the thickness. For shell sections integrated during the analysis, you can define the number of integration points through the thickness. The default is five for Simpson's rule and three for Gauss quadrature. For general shell sections, output can be obtained at three section points.

For a composite section the total number of section points is defined by adding the number of integration points per layer for all of the layers. For shell sections integrated during the analysis, you can define the number of integration points per layer. The default is three for Simpson's rule and two for Gauss quadrature. For general shell sections, the number of section points for output per layer is three.

Default Output Points

In Abaqus/Standard the default output points through the thickness of a shell section are the points that are on the bottom and top surfaces of the shell section (for integration with Simpson's rule) or the points that are closest to the bottom and top surfaces (for Gauss quadrature). For example, if five integration points are used through a single layer shell, output will be provided for section points 1 (bottom) and 5 (top).

In Abaqus/Explicit all section points through the thickness of a shell section are written to the results file for element output requests.

Choosing a Shell Element

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Shell Elements*
- *Three-Dimensional Conventional Shell Element Library*
- *Continuum Shell Element Library*
- *Axisymmetric Shell Element Library*
- *Axisymmetric Shell Elements with Nonlinear, Asymmetric Deformation*
- *Creating homogeneous shell sections*
- *Creating composite shell sections*

Overview

The Abaqus/Standard shell element library includes:

- elements for three-dimensional shell geometries;
- elements for axisymmetric geometries with axisymmetric deformation;
- elements for axisymmetric geometries with general deformation that is symmetric about one plane;
- elements for stress/displacement, heat transfer, and fully coupled temperature-displacement analysis;
- general-purpose elements, as well as elements specifically suitable for the analysis of “thick” or “thin” shells;
- general-purpose, three-dimensional, first-order elements that use reduced or full integration;
- elements that account for finite membrane strain;
- elements that use five degrees of freedom per node where possible, as well as elements that always use six degrees of freedom per node; and
- continuum shell elements.

The Abaqus/Explicit shell element library includes:

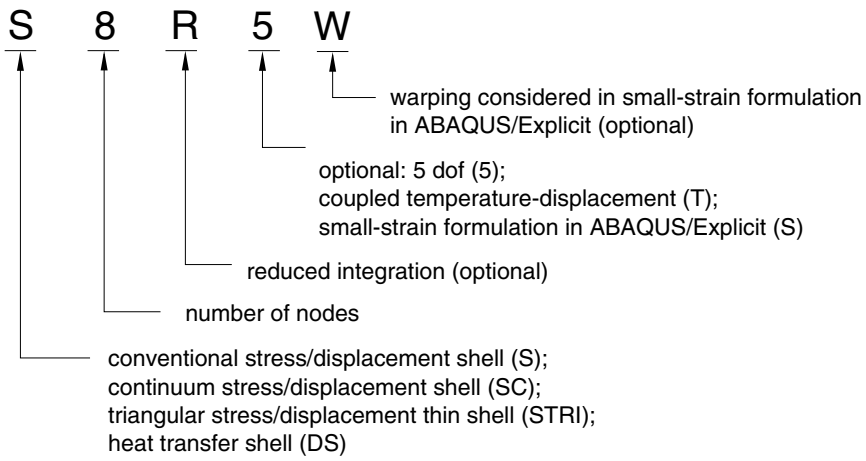
- general-purpose three-dimensional elements to model “thick” or “thin” shells that account for finite membrane strains;
- small-strain elements;
- fully coupled temperature-displacement analysis elements;
- an element for axisymmetric geometries with axisymmetric deformation; and
- continuum shell elements.

Naming Convention

The naming convention for shell elements depends on the element dimensionality.

Three-Dimensional Shell Elements

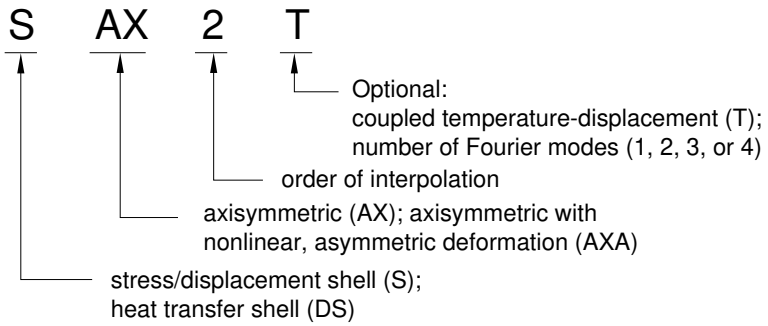
Three-dimensional shell elements in Abaqus are named as follows:



For example, S4R is a 4-node, quadrilateral, stress/displacement shell element with reduced integration and a large-strain formulation; and SC8R is an 8-node, quadrilateral, first-order interpolation, stress/displacement continuum shell element with reduced integration.

Axisymmetric Shell Elements

Axisymmetric shell elements in Abaqus are named as follows:



For example, DSAX1 is an axisymmetric, heat transfer shell element with first-order interpolation.

Conventional Stress/Displacement Shell Elements

The conventional stress/displacement shell elements in Abaqus can be used in three-dimensional or axisymmetric analysis. In Abaqus/Standard they use linear or quadratic interpolation and allow mechanical and/or thermal (uncoupled) loading; in Abaqus/Explicit they use linear interpolation and allow mechanical loading. These elements can be used in static or dynamic procedures. Some elements include the effect of transverse shear deformation and thickness change, while others do not. Some elements allow large rotations and finite membrane deformation, while others allow large rotations but small strains.

Interpolation of Temperature and Field Variables in Stress/Displacement Shell Elements

The value of temperatures at the integration locations in the surface of the shell used to compute the thermal stresses depends on whether first-order or second-order elements are used. An average temperature is used at the integration location in linear elements so that the thermal strain is constant throughout the shell surface. A linearly varying temperature distribution is used in higher-order shell elements. Field variables in stress/displacement shell elements are interpolated the same way as temperatures.

Stress/Displacement Continuum Shell Elements

The stress/displacement continuum shell elements in Abaqus can be used in three-dimensional analysis. Continuum shells discretize an entire three-dimensional body, unlike conventional shells which discretize a reference surface (see *About Shell Elements*). These elements have displacement degrees of freedom only, use linear interpolation, and allow mechanical and/or thermal (uncoupled) loading for static and dynamic procedures. The continuum shell elements are general-purpose shells that allow finite membrane deformation and large rotations and, thus, are suitable for nonlinear geometric analysis. These elements include the effects of transverse shear deformation and thickness change.

Continuum shell elements employ first-order layer-wise composite theory, and estimate through-thickness section forces from the initial elastic moduli. Unlike conventional shells, continuum shell elements can be stacked to provide more refined through-thickness response. Stacking continuum shell elements allows for a richer transverse shear stress and force prediction.

Although continuum shell elements discretize a three-dimensional body, care should be taken to verify whether the overall deformation sustained by these elements is consistent with their layer-wise plane stress assumption; that is, the response is bending dominated and no significant thickness change is observed (i.e., approximately less than 10% thickness change). Otherwise, regular three-dimensional solid elements (*Three-Dimensional Solid Element Library*) should be used. Furthermore, the thickness strain mode may yield a small stable time increment for thin continuum shell elements in Abaqus/Explicit (see *Shell Section Behavior*).

For models of thin structures that require a three-dimensional constitutive material behavior, the continuum solid shell (CSS8) element is recommended (see *Continuum Solid Shell Elements*).

Coupled Temperature-Displacement Continuum Shell Elements

The coupled temperature-displacement continuum shell elements in Abaqus have continuum shell geometry and use linear interpolation for the geometry and displacements. The temperature is interpolated linearly as well. The thermal formulation is similar to that used for three-dimensional coupled temperature-displacement solid elements with reduced integration, for which the temperature variation is trilinear (see *Solid (Continuum) Elements*). The temperatures at the section points through the thickness are interpolated linearly from the temperatures at the nodes.

Heat Transfer Shell Elements

These elements, available only in Abaqus/Standard and only with conventional shell element geometry, are intended to model heat transfer in shell-type structures. They provide the values of temperature at a number of points through the thickness at each shell node. This output can be input directly to the equivalent stress analysis shell element for sequentially coupled thermal-stress analysis (*Sequentially Coupled Thermal-Stress Analysis*).

Temperature Variation through the Shell Thickness

The temperature variation through the thickness of a shell is always assumed to be piecewise quadratic irrespective of the ratio of the thickness to in-plane dimensions in a three-dimensional shell or the ratio of the thickness to radius in an axisymmetric shell. The interpolation on the reference surface of the shell is the same as that of the corresponding stress elements. For shell sections integrated during the analysis (*Using a Shell Section Integrated during the Analysis to Define the Section Behavior*) you can specify the number of section points used for cross-section integration and thickness-direction temperature interpolation at each node. Only Simpson's rule can be used for integration through the shell thickness.

The temperature on the bottom surface of the shell (the surface in the negative direction along the shell normal—see *Defining the Initial Geometry of Conventional Shell Elements*) is degree of freedom 11. The temperature on the top surface is degree of freedom $\mathbf{10} + \boldsymbol{n_s}$. A maximum of 20 temperature degrees of freedom can exist at a node. For a single-layer shell $\boldsymbol{n_s}$ is the total number of integration points used through the shell section. If a single section point is used for the cross-section integration, there is no temperature variation through

the thickness of the shell and the temperature of the entire shell cross-section is degree of freedom 11. For a multi-layered shell the temperature at the top of each layer is the same as the temperature at the bottom of the next layer. Therefore,

$$n_s = 1 + \sum_{l=1}^{\text{layers}} (n_l - 1),$$

where n_l ($n_l > 1$) is the number of integration points used in layer l. If n_l=1, n_s is equal to the number of composite layers. In this case, there is no temperature variation through the thickness of the shell, and the temperature of the entire composite is degree of freedom 11. The internal energy storage and heat conduction terms for shells are integrated in the same way as in the corresponding continuum elements (see *Solid (Continuum) Elements*).

Using Shells in a Thermal-Stress Analysis

To use the temperatures that are saved in the Abaqus/Standard results file directly as input to a thermal-stress analysis, the mesh and the specification of the number of temperature points in the shell sections must be the same in the heat transfer and the stress analysis models. In addition, multi-layered heat transfer shell elements must have the same number of integration points in each layer.

Coupled Temperature-Displacement Shell Elements

The coupled temperature-displacement shell elements available in Abaqus have conventional shell element geometry and use linear or quadratic interpolation for the geometry and displacements. The temperature is interpolated linearly from the corner or end nodes; the lower-order temperature interpolation in quadratic shells is chosen to give the same interpolation order for thermal strain, which is proportional to temperature, as for total strain. All terms in the governing equations are integrated in the reference surface of the shell using a conventional Gauss scheme; Simpson's rule is used to integrate through the shell thickness.

Temperature Variation through the Shell Thickness

The temperature variation through the shell thickness is assumed to be piecewise quadratic and is interpolated from temperatures at a series of points through the thickness of the shell at each node. The number of temperature values to be used at each node is determined by the number of integration points that you specify in the shell section definition (see *Defining the Shell Section Integration*). Up to a maximum of 20 temperature values are stored as degrees of freedom 11, 12, 13, etc. (up to degree of freedom 30) in a manner that is identical to that used for heat transfer shell elements (see *Heat Transfer Shell Elements* above).

"Thick" Versus "Thin" Conventional Shell Elements

Abaqus includes general-purpose, conventional shell elements as well as conventional shell elements that are valid for thick and thin shell problems. See below for a discussion of what constitutes a "thick" or "thin" shell problem. This concept is relevant only for elements with displacement degrees of freedom.

The general-purpose, conventional shell elements provide robust and accurate solutions to most applications and will be used for most applications. However, in certain cases, for specific applications in Abaqus/Standard, enhanced performance may be obtained with the thin or thick conventional shell elements; for example, if only small strains occur and five degrees of freedom per node are desired.

The continuum shell elements can be used for any thickness; however, thin continuum shell elements may result in a small stable time increment in Abaqus/Explicit.

General-Purpose Conventional Shell Elements

These elements allow transverse shear deformation. They use thick shell theory as the shell thickness increases and become discrete Kirchhoff thin shell elements as the thickness decreases; the transverse shear deformation becomes very small as the shell thickness decreases.

Element types S3/S3R, S3RS, S3RT, S4, S4R, S4RS, S4RSW, S4RT, SAX1, SAX2, SAX2T, SC6R, and SC8R are general-purpose shells.

Thick Conventional Shell Elements

In Abaqus/Standard thick shells are needed in cases where transverse shear flexibility is important and second-order interpolation is desired. When a shell is made of the same material throughout its thickness, this occurs when the thickness is more than about 1/15 of a characteristic length on the surface of the shell, such as the distance between supports for a static case or the wavelength of a significant natural mode in dynamic analysis.

Abaqus/Standard provides element types S8R and S8RT for use only in thick shell problems.

Thin Conventional Shell Elements

In Abaqus/Standard thin shells are needed in cases where transverse shear flexibility is negligible and the Kirchhoff constraint must be satisfied accurately (i.e., the shell normal remains orthogonal to the shell reference surface). For homogeneous shells this occurs when the thickness is less than about 1/15 of a characteristic length on the surface of the shell, such as the distance between supports or the wave length of a significant eigenmode. However, the thickness may be larger than 1/15 of the element length.

Abaqus/Standard has two types of thin shell elements: those that solve thin shell theory (the Kirchhoff constraint is satisfied analytically) and those that converge to thin shell theory as the thickness decreases (the Kirchhoff constraint is satisfied numerically).

- The element that solves thin shell theory is STRI3. STRI3 has six degrees of freedom at the nodes and is a flat, faceted element (initial curvature is ignored). If STRI3 is used to model a thick shell problem, the element will always predict a thin shell solution.
- The elements that impose the Kirchhoff constraint numerically are S4R5, STRI65, S8R5, S9R5, SAXA1*n*, and SAXA2*n*. These elements should not be used for applications in which transverse shear deformation is important. If these elements are used to model a thick shell problem, the elements may predict inaccurate results.

Finite-Strain Versus Small-Strain Shell Elements

Abaqus has both finite-strain and small-strain shell elements. This concept is relevant only for elements with displacement degrees of freedom.

Finite-Strain Shell Elements

Element types S3/S3R, S4, S4R, S4T, SAX1, SAX2, SAX2T, SAXA1*n*, and SAXA2*n* account for finite membrane strains and arbitrarily large rotations; therefore, they are suitable for large-strain analysis. The underlying formulation is described in *Axisymmetric shell elements*, *Finite-strain shell element formulation*, and *Axisymmetric shell element allowing asymmetric loading*.

Continuum shell elements SC6R and SC8R account for finite membrane strains, arbitrary large rotation, and allow for changes in thickness, making them suitable for large-strain analysis. Computation of the change in thickness is based on the element nodal displacements, which in turn are computed from an effective elastic modulus defined at the beginning of an analysis.

Small-Strain Shell Elements

In Abaqus/Standard the three-dimensional "thick" and "thin" element types STRI3, S4R5, STRI65, S8R, S8RT, S8R5, and S9R5 provide for arbitrarily large rotations but only small strains. The change in thickness with deformation is ignored in these elements.

In Abaqus/Explicit element types S3RS, S4RS, and S4RSW are provided for shell problems with small membrane strains and arbitrarily large rotations. Many impact dynamics analyses fall within this class of problems, including those of shell structures undergoing large-scale buckling behavior but relatively small amounts of membrane stretching and compression. Although solution accuracy may degrade as membrane strains become large, the small-strain shell elements in Abaqus/Explicit provide a computationally efficient alternative to the finite-membrane-strain elements for appropriate applications. The underlying formulation is described in *Small-strain shell elements in Abaqus/Explicit*.

Change of Shell Thickness

Thickness change is considered only in geometrically nonlinear analyses. For conventional shells, stress in the thickness direction is zero and the strain results only from the Poisson's effect. For continuum shells, the stress in the thickness direction may not be zero and may cause additional strain beyond that due to Poisson's effect. The thickness strain due to Poisson's effect is referred as the "Poisson strain," and any additional strain beyond the "Poisson strain" is referred to as the "effective thickness strain."

For shell elements in Abaqus/Explicit defined by integrating the section during the analysis, the Poisson strain is calculated by enforcing the plane stress condition either at the individual material points in the section and then integrating the Poisson strain from these material points, or at the integration station for the whole section using a "section Poisson's ratio." For shell elements in Abaqus/Standard only the section Poisson's ratio method is available. For shell elements defined by general shell sections, only the section Poisson's ratio method is applicable.

See *Defining the Poisson Strain in Shell Elements in the Thickness Direction* for details.

Thickness Direction Stress in Continuum Shell Elements

The thickness direction stress is computed by penalizing the effective thickness strain with a constant "thickness modulus." The thickness modulus used for a single layer shell element with an elastic or elastic-plastic material is twice the in-plane elastic shear modulus. In the case of a composite shell with each layer either an elastic or elastic-plastic material, the thickness modulus is computed as the thickness-weighted harmonic mean of the contributions from the individual layers:

$$1/E_{eff} = \sum_{i}^{n} r_i / E_{eff}^{i},$$

where E_{eff} is the thickness modulus, i is the layer index, n is the number of layers, r_i is the relative thickness of layer i, $(0 < r_i < 1)$, and E_{eff}^{i} is twice the initial in-plane elastic shear modulus based on the material definition for layer i in the initial configuration.

See *Defining the Thickness Modulus in Continuum Shell Elements* and *Defining the Thickness Modulus in Continuum Shell Elements* for details.

Five Degree of Freedom Shells Versus Six Degree of Freedom Shells

Two types of three-dimensional conventional shell elements are provided in Abaqus/Standard: ones that use five degrees of freedom (three displacement components and two in-surface rotation components) where possible and ones that use six degrees of freedom (three displacement components and three rotation components) at all nodes.

The elements that use five degrees of freedom (S4R5, STRI65, S8R5, S9R5) can be more economical. However, they are available only as "thin" shells (they cannot be used as "thick" shells) and cannot be used for finite-strain applications (although they model large rotations with small strains accurately). In addition, output for the five degree of freedom shell elements is restricted as follows:

- At nodes that use the two in-surface rotation components, the values of these in-surface rotations are not available for output.
- When output variable NFORC is requested, moments corresponding to the in-surface rotations are not available for output.

When five degree of freedom shell elements are used, Abaqus/Standard will automatically switch to using three global rotation components at any node that:

- has kinematic boundary conditions applied to rotational degrees of freedom,
- is used in a multi-point constraint (*General Multi-Point Constraints*) that involves rotational degrees of freedom,
- is shared with a beam element or a shell element that uses the three global rotation components at all nodes,
- is on a fold line in the shell (that is, on a line where shells with different surface normals come together), or
- is loaded with moments.

In all elements that use three global rotation components at all nodes (whether activated as described above or always present), a singularity exists at any node where the surface is assumed to be continuously curved: three rotation components are used, but only two are actively associated with stiffness. A small stiffness is associated with the rotation about the normal to avoid this difficulty. The default stiffness values used are sufficiently small such that the artificial energy content is negligible. In some rare cases this stiffness may need to be altered. You can define a scaling factor for this stiffness, as described in *Using a Shell Section Integrated during the Analysis to Define the Section Behavior* and *Using a General Shell Section to Define the Section Behavior*.

Reduced Integration

Many shell element types in Abaqus use reduced (lower-order) integration to form the element stiffness. The mass matrix and distributed loadings are still integrated exactly. Reduced integration usually provides more accurate results (provided the elements are not distorted or loaded in in-plane bending) and significantly reduces running time, especially in three dimensions.

When reduced integration is used with first-order (linear) elements, hourglass control is required. Therefore, when using first-order reduced-integration elements, you must check if hourglassing is occurring; if it is, a finer mesh may be required or concentrated loads must be distributed over multiple nodes. The second-order reduced-integration elements available in Abaqus/Standard generally do not have the same difficulty and are recommended in cases when the solution is expected to be smooth. First-order elements are recommended when large strains or very high strain gradients are expected.

Specifying Section Controls for Shell Elements

In Abaqus/Standard you can specify nondefault hourglass control parameters for shell elements. In Abaqus/Explicit you can specify second-order accuracy in the element formulation, nondefault hourglass control parameters for S4R, S4RS, and S4RSW elements, or deactivate the drill constraint for S3RS and S4RS elements. See *Section Controls* for more information.

Input File Usage: Use the following options in Abaqus/Standard:

**SHELL SECTION* or **SHELL GENERAL SECTION*
**HOURGLASS STIFFNESS*

Use one of the following options in Abaqus/Explicit:

**SHELL SECTION*, CONTROLS=*name*
**SHELL GENERAL SECTION*, CONTROLS=*name*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Element Controls**

Modeling Issues

A number of modeling issues must be considered when using shell elements.

Using S3/S3R and S3RS Elements

Both S3 and S3R refer to the same 3-node triangular shell element. This element is a degenerated version of S4R that is fully compatible with S4R and, in Abaqus/Standard, S4.

Element S3RS, available in Abaqus/Explicit, is a degenerated version of S4RS that is fully compatible with S4RS.

S3/S3R and S3RS provide accurate results in most loading situations. However, because of their constant bending and membrane strain approximations, high mesh refinement may be required to capture pure bending deformations or solutions to problems involving high strain gradients. A consequence of the degenerated element formulation is that the solution changes slightly when the element connectivity is permuted.

Degenerating Elements

Element types S4, S4R, S4R5, S4RS, S8R5, and S9R5 can be degenerated to triangles. However, for elements S4 (element S4 degenerated to a triangle may exhibit overly stiff response in membrane deformation), S4R, and S4RS it is recommended that S3R and S3RS be used instead.

The quarter-point technique (moving the midside nodes to the quarter points to give a $1/\sqrt{r}$ singularity for elastic fracture mechanics applications) can be used with the quadratic element types S8R5 and S9R5 (see *Element Definition*). The accuracy of the element is very significantly reduced when it is degenerated to a triangle; therefore, this is *not* recommended except for special applications, such as fracture.

Element types S8R and S8RT cannot be degenerated to triangles. Element types DS4 and DS8 can be degenerated to triangles, but it is recommended that DS3 and DS6 elements be used instead.

Modeling with Continuum Shell Elements

Continuum shell elements are similar to continuum solids from a modeling point of view. The element geometries for the SC6R and SC8R elements are a triangular prism and hexahedron, respectively, with displacement degrees of freedom only.

Continuum shell elements must be oriented correctly, since these elements have a thickness direction associated with them. See *About Shell Elements* for further details on element connectivity and orientation.

When classical shell structures (structures in which only the midsurface geometry and kinematic constraints are provided) are analyzed, care must be taken that appropriate moments and rotations are specified. For example, a moment may be applied as a force-couple system at the corresponding nodes on the top and bottom faces. A rotation boundary condition may be specified through a kinematic constraint to yield the appropriate displacement boundary conditions on the edge of the continuum shell.

Continuum shell elements can be connected directly to first-order continuum solids without any kinematic transition. An appropriate kinematic transition needs to be provided when conventional shell elements are connected to continuum shell elements to correctly transfer the moment/rotation at the reference surface of a

conventional shell. Such a transition can be defined with a shell-to-solid coupling constraint or any other kinematic constraint, such as a surface-based coupling constraint, a multi-point constraint, or a linear constraint equation.

Using the SC6R Element

The SC6R element is a degenerated version of the SC8R element. The SC6R element provides accurate results in most loading situations. However, because of its constant bending and membrane strain approximations, high mesh refinement may be required to capture pure bending deformations or solutions to problems involving high strain gradients.

Modeling Contact with Continuum Shell Elements

Continuum shell elements, SC6R and SC8R, allow two-sided contact with changes in the thickness and are thus suitable for modeling contact.

Stable Time Increment in Abaqus/Explicit

In Abaqus/Explicit the element stable time increment can be controlled by the continuum shell element thickness, particularly for thin shell applications. This may increase significantly the number of increments taken to complete the analysis when compared to the same problem modeled with conventional shell elements. The small stable time increment size may be mitigated by specifying a lower stiffness in the thickness direction when appropriate.

Limitations with Continuum Shell Elements

Continuum shell elements cannot be used with the hyperfoam material definitions, nor can they be used with general shell sections where the section stiffness is provided directly.

Modeling a “Sandwich” Shell

For a “sandwich” shell, in which parts of the cross-section are made of a softer material (especially when the layers are nonisotropic so that some layers are weak in particular directions), the transverse shear flexibility can be important even when the shell is rather thin. Use of general-purpose shell elements or stacking continuum shell elements is recommended in such cases. See *Shell Section Behavior* for a discussion of transverse shear stiffness in shell elements.

Modeling Bending of a Thin Curved Shell in Abaqus/Standard

In Abaqus/Standard curved elements (STRI65, S8R5, S9R5) are preferable for modeling bending of a thin curved shell.

Element type STRI3 is a flat facet element. If this element is used to model bending of a curved shell, a dense mesh may be required to obtain accurate results.

Modeling Buckling of Doubly Curved Shells in Abaqus/Standard

Element type S8R5 may give inaccurate results for buckling problems of doubly curved shells due to the fact that the internally defined center node may not be positioned on the actual shell surface. Element type S9R5 should be used instead.

Using S8R5 in Contact Analyses

Element type S8R5 is converted automatically to element type S9R5 if a secondary surface in a contact pair is attached to the element.

Applying Moments to S9R5 Elements

Moments should not be applied to the center node of S9R5 elements.

Using S4 Elements

Element type S4 is a fully integrated, general-purpose, finite-membrane-strain shell element. The element's membrane response is treated with an assumed strain formulation that gives accurate solutions to in-plane bending problems, is not sensitive to element distortion, and avoids parasitic locking.

Element type S4 does not have hourglass modes in either the membrane or bending response of the element; hence, the element does not require hourglass control. The element has four integration locations per element compared with one integration location for S4R, which makes the element computationally more expensive. S4 is compatible with both S4R and S3R. S4 can be used for problems prone to membrane- or bending-mode hourglassing, in areas where greater solution accuracy is required, or for problems where in-plane bending is expected. In all of these situations S4 will outperform element type S4R. S4 cannot be used with the hyperelastic or hyperfoam material definitions in Abaqus/Standard.

Defining the Initial Geometry of Conventional Shell Elements

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Shell Elements*
- *Assigning a section*
- *Assigning shell/membrane normal directions*

Overview

The initial shell geometry:

- is defined by initial normal directions, which can be user-defined or calculated by Abaqus;
- requires that sufficient mesh refinement be used so that the discretized surface accurately represents the actual surface; and
- can include an offset of the reference surface from the shell's midsurface.

Defining Nodal Normals

This discussion applies to conventional shell elements only. The normals of a continuum shell element are defined by the position of the top and bottom nodes along the shell corner edge (see *About Shell Elements*).

Shell normals can be defined by giving the direction cosines of the normal to the surface at all nodes attached to shell elements. These direction cosines can be entered as the fourth, fifth, and sixth coordinates of each node definition or in a user-specified normal definition, as described below; see *Normal Definitions at Nodes* for more information. If the user-defined normal differs from the midsurface normal by more than 20°, a warning message is issued to the data (`.dat`) file. However, if the angle is more than 160°, the direction of the midsurface normal is reversed and no warning message is issued. An additional warning message is issued if the nodal normal deviates more than 10° from the average element normal.

Specifying the same normal at a node for all shell elements attached to the node represents a smooth shell surface at the node. Define a user-specified normal to introduce a fold line.

If the normals are not defined as part of the node definition or by a user-specified normal, Abaqus will calculate the normal using the algorithm given below. Since the only information available for this calculation is the nodal coordinates, it may not define the normal directions accurately. Accurate definition can be important on edges of the model, especially if they are also symmetry planes, or on lines where the curvature of the shell changes discontinuously.

The normal direction at a node is needed for temperature input and nodal stress output. The direction is taken from the definitions below for the elements adjacent to the nodes. If this leads to a conflict at a node, the positive normal direction used at that node will be the one defined by the lowest numbered element at the node.

Calculation of Average Nodal Normals by Abaqus

If the nodal normal is not defined as part of the node definition, element normal directions at the node are calculated for all shell and beam elements for which a user-specified normal is not defined (the “remaining” elements). For shell elements the normal direction is orthogonal to the shell midsurface, as described in *About Shell Elements*. For beam elements the normal direction is the second cross-section direction, as described in *Beam Element Cross-Section Orientation*.

The following algorithm is then used to obtain an average normal (or multiple averaged normals) for the remaining elements that need a normal defined:

1. If a node is connected to more than 30 remaining elements, no averaging occurs and each element is assigned its own normal at the node. The first nodal normal is stored as the normal defined as part of the node definition. Each subsequent normal is stored as a user-specified normal.
2. If a node is shared by 30 or fewer remaining elements, the normals for all the elements connected to the node are computed. Abaqus takes one of these elements and puts it in a set with all the other elements that have normals within 20° of it. Then:
   a. Each element whose normal is within 20° of the added elements is also added to this set (if it is not yet included).
   b. This process is repeated until the set contains for each element in the set all the other elements whose normals are within 20°.
   c. If all the normals in the final set are within 20° of each other, an average normal is computed for all the elements in the set. If any of the normals in the set are more than 20° out of line from even a single other normal in the set, no averaging occurs for elements in the set and a separate normal is stored for each element.
   d. This process is repeated until all the elements connected to the node have had normals computed for them.
   e. The first nodal normal is stored as the normal defined as part of the node definition. Each subsequently generated nodal normal is stored as a user-specified normal.

This algorithm ensures that the nodal averaging scheme has no element order dependence. A simple example illustrating this process is included below.

Example: Shell Normal Averaging

Consider the three element model in *Figure 1*. Elements 1, 2, and 3 share a common node, node 10, with no user-specified normal defined.

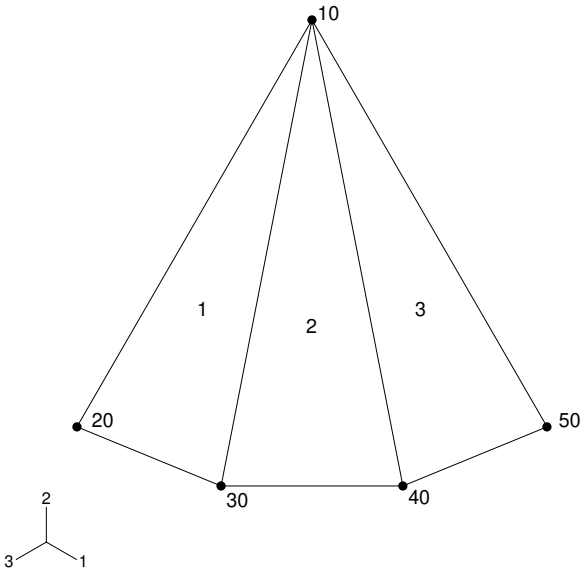


Figure 1: Three element example for nodal averaging algorithm.

In the first scenario, suppose that at node 10 the normal for element 2 is within 20° of both elements 1 and 3, but the normals for elements 1 and 3 are not within 20° of each other. In this case, each element is assigned its own normal: one is stored as part of the node definition and two are stored as user-specified normals.

In the second scenario, suppose that at node 10 the normal for element 2 is within 20° of both elements 1 and 3 and the normals for elements 1 and 3 are within 20° of each other. In this case, a single average normal for elements 1, 2, and 3 would be computed and stored as part of the node definition.

In the last scenario, suppose that at node 10 the normal for element 2 is within 20° of element 1 but the normal of element 3 is not within 20° of either element 1 or 2. In this case, an average normal is computed and stored for elements 1, and 2 and the normal for element 3 is stored by itself: one is stored as part of the node definition and the other is stored as a user-specified normal.

Meshing Concerns

In a coarse mesh this algorithm may introduce fold lines where the shell is smooth, or it may create a smooth shell where there should be a fold if the angle of the fold line is less than 20°. Difficulties in large-displacement shell analysis are sometimes caused by false fold lines introduced by coarse meshing. To model a smooth shell, the mesh should be refined enough to create unique nodal normals. To model plates or shells with fold lines, you should define user-specified normals.

Verifying the Normal Definitions

Normal definitions can be checked by examining the analysis input file processor output. The direction cosines of the reference normal associated with a node are listed under the `NODE DEFINITIONS` output in the data (`.dat`) file. User-specified normals are listed under the `NORMAL DEFINITIONS` output in the data file.

Offset: Reference Surface Versus Midsurface

This discussion applies to conventional shell elements only. Continuum shell elements define a top and bottom surface around the structural body being modeled. The notion of a shell reference surface is not applicable for these types of elements.

The reference surface for conventional shell elements is defined by the shell's nodes and normal definitions. When modeling with shell elements, the reference surface is typically coincident with the shell's midsurface. However, many situations arise in which it is more convenient to define the reference surface as offset from the shell's midsurface. For example, CAD surfaces usually represent either the top or bottom surface of the shell. In this case it may be easier to define the reference surface to be coincident with the CAD surface and, therefore, offset from the shell's midsurface.

Shell offsets can also be used to define a more precise surface geometry for contact problems where shell thickness is important. Another situation where the offset from the midsurface may be important is when a shell with continuously varying thickness is modeled. In this case if one surface of the shell is smooth while the other surface is rough, as in some aircraft structures, using the smooth surface as the reference surface, with an offset of half the shell's thickness from the midsurface, will represent the physical geometry more accurately. The use of the midsurface as the reference surface for this case is much more complicated and may result in an inaccurate model.

You can introduce offsets in the section definitions for both shell sections integrated during the analysis and general shell sections. The offset value is defined as a fraction of the shell thickness measured from the shell's midsurface to the shell's reference surface. See *Using a Shell Section Integrated during the Analysis to Define the Section Behavior* and *Using a General Shell Section to Define the Section Behavior* for details.

The degrees of freedom for the shell are associated with the reference surface. The element's area and all kinematic quantities are calculated there. Any loading in the plane of the reference surface will, therefore, cause both membrane forces and bending moments when a nonzero offset value is used. Large offset values for curved shells may also lead to a surface integration error, affecting the stiffness, mass, and rotary inertia for the shell

section. For stability purposes Abaqus/Explicit also automatically scales the rotary inertia used for shell elements by a factor proportional to the offset squared, which may result in errors for large offsets. When a large offset from the shell's midsurface is necessary, use multi-point constraints instead (see *General Multi-Point Constraints*).

Shell Section Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Shell Elements*
- *Using a Shell Section Integrated during the Analysis to Define the Section Behavior*
- *Using a General Shell Section to Define the Section Behavior*
- **SHELL GENERAL SECTION*
- **SHELL SECTION*
- *Creating homogeneous shell sections*
- *Creating composite shell sections*

Overview

The shell section behavior:

- may or may not require numerical integration over the section;
- can be linear or nonlinear; and
- can be homogeneous or composed of layers of different material.

Methods for Defining the Shell Section Behavior

Two methods are provided to define the cross-sectional behavior of a shell.

- Linear moment-bending and force-membrane strain relationships can be defined by using a general shell section (see *Using a General Shell Section to Define the Section Behavior*). In this case all calculations are done in terms of section forces and moments.

  In Abaqus/Standard when section properties are given directly (i.e., the section is not associated with one or more material definitions), strains and stresses are not available for output. However, when section properties are specified by one or more elastic material layers, strains and stresses are available when requested for output. In Abaqus/Explicit stresses and strains are not available for output at the section points whenever a general shell section is used; only section forces, section moments, and section strains are available for output.

  In Abaqus/Standard nonlinear behavior of the shell section, formulated in terms of forces and moments, can be defined by using a general shell section in conjunction with user subroutine *UGENS*.
- Alternatively, a shell section integrated during the analysis (see *Using a Shell Section Integrated during the Analysis to Define the Section Behavior*) allows the cross-sectional behavior to be calculated by numerical integration through the shell thickness, thus providing complete generality in material modeling. With this type of section any number of material points can be defined through the thickness and the material response can vary from point to point.

Both general shell sections and shell sections integrated during the analysis allow layers of different materials, in different orientations, to be used through the cross-section. In these cases the section definition provides the shell thickness, material, and orientation per layer.

For conventional shell elements you can specify an offset of the reference surface from the shell's midsurface when the section properties are specified by one or more material layers. When the section properties are given directly, you cannot directly specify an offset; however, an offset can be included implicitly in the section properties. A nonzero offset cannot be specified for continuum shell elements. If a nonzero offset is specified for a continuum shell element, an error message is issued during input file preprocessing.

Determining Whether to Use a Shell Section Integrated during the Analysis or a General Shell Section

When a shell section integrated during the analysis (see *Using a Shell Section Integrated during the Analysis to Define the Section Behavior*) is used, Abaqus uses numerical integration through the thickness of the shell to calculate the section properties. This type of shell section is generally used with nonlinear material behavior in the section. It must be used with shells that provide for heat transfer, since general shell sections do not allow the definition of heat transfer properties.

Use a general shell section (see *Using a General Shell Section to Define the Section Behavior*) if the response of the shell is linear elastic and its behavior is not dependent on changes in temperature or predefined field variables or, in Abaqus/Standard, if nonlinear behavior in terms of forces and moments is to be defined in user subroutine *UGENS*.

Transverse Shear Stiffness

For all shell elements in Abaqus/Standard that use transverse shear stiffness and for the finite-strain shell elements in Abaqus/Explicit, the transverse shear stiffness is computed by matching the shear response for the shell to that of a three-dimensional solid for the case of bending about one axis. For the small-strain shell elements in Abaqus/Explicit the transverse shear stiffness is based on the effective shear modulus.

Transverse Shear Stiffness for Shell Elements in Abaqus/Standard and Finite-Strain Shell Elements in Abaqus/Explicit

In all shell elements in Abaqus/Standard that are valid for thick shell problems or that enforce the Kirchhoff constraint numerically (i.e., all shell elements except STRI3) and in the finite-strain shell elements in Abaqus/Explicit (S3R, S4, S4R, SAX1, SC6R, and SC8R), Abaqus computes the transverse shear stiffness by matching the shear response for the case of the shell bending about one axis, using a parabolic variation of transverse shear stress in each layer. The approach is described in *Transverse shear stiffness in composite shells and offsets from the midsurface* and generally provides a reasonable estimate of the shear flexibility of the shell. It also provides estimates of interlaminar shear stresses in composite shells. In calculating the transverse shear stiffness, Abaqus assumes that the shell section directions are the principal bending directions (bending about one principal direction does not require a restraining moment about the other direction). For composite shells with orthotropic layers that are not symmetric about the shell midsurface, the shell section directions may not be the principal bending directions. In such cases the transverse shear stiffness is a less accurate approximation and will change if different shell section directions are used. Abaqus computes the transverse shear stiffness only once at the beginning of the analysis based on initial elastic properties given in the model data. Initial temperature and field variable dependencies are considered. Any changes to the transverse shear stiffness that occur due to changes in the material stiffness during the analysis are ignored.

Axisymmetric shell elements SAX1 and SAX2; three-dimensional shell elements S3/S3R, S4, S4R, S8R, and S8RT; and continuum shell elements SC6R and SC8R are based on a first-order shear deformation theory. Other shell elements—such as S4R5, S8R5, S9R5, STRI65, and SAXA*mn*—use the transverse shear stiffness to enforce the Kirchhoff constraints numerically in the thin shell limit. The transverse shear stiffness is not relevant for shells without displacement degrees of freedom and is relevant only for element type STRI3 as part of a penalty term used to control drill rotations. Although element type S4 has four integration points, the transverse shear calculation is assumed constant over the element. A higher resolution of the transverse shear may be obtained by stacking continuum shell elements.

For most shell sections, including layered composite or sandwich shell sections, Abaqus calculates the transverse shear stiffness values required in the element formulation. You can override these default values (see *Defining the Transverse Shear Stiffness* for shell sections and *Defining the Transverse Shear Stiffness* for general shell sections). In some cases the default shear stiffness values are not calculated if estimates of the shear moduli are unavailable during the preprocessing stage of input; for example, when the section behavior is defined in user subroutine *UGENS* in Abaqus/Standard. You must define the transverse shear stiffnesses in such cases. When a shell section's material behavior is defined by a user subroutine (for example, *UMAT*, *UHYPEL*, *UHYPER*, or

VUMAT), you must define the transverse shear modulus for the material (see *Defining the Elastic Transverse Shear Modulus*) or the transverse shear stiffness for the section.

Transverse Shear Stiffness Definition

The transverse shear stiffness of the section of a shear flexible shell element is defined in Abaqus as

$$\overline{K}^{ts}_{\alpha\beta} = f_p K^{ts}_{\alpha\beta},$$

where

$\overline{K}^{ts}_{\alpha\beta}$

are the components of the section shear stiffness ($\alpha, \beta = 1, 2$ refer to the default surface directions on the shell, as defined in *Conventions*, or to the local directions associated with the shell section definition);

f_p

is a dimensionless factor that is used to prevent the shear stiffness from becoming too large in thin shells; and

$K^{ts}_{\alpha\beta}$

is the actual shear stiffness of the section (calculated by Abaqus or user-defined).

You can specify all three shear stiffness terms (K^{ts}_{11}, K^{ts}_{22}, and $K^{ts}_{12} = K^{ts}_{21}$); otherwise, they will take the default values defined below. The dimensionless factor f_p is always included in the calculation of transverse shear stiffness, regardless of the way $K^{ts}_{\alpha\beta}$ is obtained. For shell elements of type S4R5, S8R5, S9R5, STRI65, or SAXA*n* the average of K^{ts}_{11} and K^{ts}_{22} is used and K^{ts}_{12} is ignored. The $K^{ts}_{\alpha\beta}$ have units of force per length.

The dimensionless factor f_p is defined as

$$f_p = 1/\left(1 + 0.25 \times 10^{-4} \frac{A}{t^2}\right),$$

where A is the area of the element and t is the thickness of the shell. When a general shell section definition not associated with one or more material definitions is used to define the shell section stiffness, the thickness of the shell, t, is estimated as

$$t = \sqrt{12 \frac{D_{44} + D_{55} + D_{66}}{D_{11} + D_{22} + D_{33}}}.$$

If you do not specify the $K^{ts}_{\alpha\beta}$, they are calculated as follows. For laminated plates and sandwich constructions the $K^{ts}_{\alpha\beta}$ are estimated by matching the elastic strain energy associated with shear deformation of the shell section with that based on piecewise quadratic variation of the transverse shear stress across the section, under conditions of bending about one axis. For unsymmetric lay-ups the coupling term K^{ts}_{12} can be nonzero.

When a general shell section is used and the section stiffness is given directly, the $K^{ts}_{\alpha\beta}$ are defined as

$$K^{ts}_{11} = K^{ts}_{22} = \left(\frac{1}{6}\left(D_{11} + D_{22}\right) + \frac{1}{3} D_{33}\right) Y, \quad K^{ts}_{12} = 0,$$

where D_{ij} is the section stiffness matrix and Y is the initial scaling modulus.

When a user subroutine (for example, *UMAT*, *UHYPEL*, *UHYPER*, or *VUMAT*) is used to define a shell element's material response, you must define the transverse shear modulus for the material or the transverse shear stiffness for the shell. The definition of an appropriate stiffness depends on the shell's material composition and its lay-up; that is, how the material is distributed through the thickness of the cross-section.

The transverse shear stiffness should be specified as the initial, linear elastic stiffness of the shell in response to pure transverse shear strains. For a homogeneous shell made of a linear, orthotropic elastic material, where the strong material direction aligns with the element's local 1-direction, the transverse shear stiffness should be

$$K_{11}^{ts} = \frac{5}{6} G_{13} t \ , \quad K_{22}^{ts} = \frac{5}{6} G_{23} t \ , \quad \text{and} \quad K_{12}^{ts} = 0.0 \, .$$

G_{13} and G_{23} are the material's shear moduli in the out-of-plane direction. The number 5/6 is the shear correction coefficient that results from matching the transverse shear energy to that for a three-dimensional structure in pure bending. For composite shells the shear correction coefficient is different from the value for homogeneous ones; see *Transverse shear stiffness in composite shells and offsets from the midsurface* for a discussion of how the effective shear stiffness for elastic materials is obtained in Abaqus.

Checking the Validity of Using Shell Theory

For linear elastic materials the slenderness ratio, $K_{\alpha\alpha} l^2 / D_{(\alpha+3)(\alpha+3)}$, where α=1 or 2 (no sum on α) and l is a characteristic length on the surface of the shell, can be used as a guideline to decide if the assumption that plane sections must remain plane is satisfied and, hence, shell theory is adequate. Generally, if

$$\frac{K_{\alpha\alpha} l^2}{D_{(\alpha+3)(\alpha+3)}} > 100,$$

shell theory will be adequate; for smaller values the membrane strains will not vary linearly through the section, and shell theory will probably not give sufficiently accurate results. The characteristic length, l, is independent of the element length and should not be confused with the element's characteristic length, L_c.

To obtain the $K_{\alpha\alpha}$ and $D_{(\alpha+3)(\alpha+3)}$, you must run a data check analysis using a composite general shell section definition. The $K_{\alpha\alpha}$ will be printed under the title "TRANSVERSE SHEAR STIFFNESS FOR THE SECTION" in the data (`.dat`) file if you request model definition data (see *Controlling the Amount of analysis input file processor Information Written to the Data File*). The $D_{\alpha\beta}$ will be printed out under the title "SECTION STIFFNESS MATRIX."

Transverse Shear Stiffness for Small-Strain Shell Elements in Abaqus/Explicit

When a shell section integrated during the analysis is used, the transverse shear stresses for the small-strain shells in Abaqus/Explicit are assumed to have a piecewise constant distribution in each layer. The transverse shear force will converge to the correct solution for single or multilayer isotropic sections and single-layer orthotropic sections. The transverse shear stiffness is approximate for multilayer orthotropic sections where convergence to the proper transverse shear behavior may not be obtained as shells become thick and principal material directions deviate from the principal section directions. The finite-strain S4R element should be used with a shell section integrated during the analysis if accurate through-thickness transverse shear stress distributions are required for the analysis of composite shells.

The same transverse shear stiffness described for the finite-strain shells is used to calculate the transverse shear force for the small-strain shells in Abaqus/Explicit when a general shell section is used. Thus, for this case the transverse shear force for multilayer composite shells will converge to the correct value for both thin and thick sections.

Drill Stiffness and Transverse Shear

Shell elements in Abaqus use a small penalty stiffness to control drill rotations. When used, the drill stiffness is proportional to the transverse shear stiffness. Drill stiffness is not needed for a five degree-of-freedom shell element unless three global rotation components are active on the element.

Bending Strain Measures

Most three-dimensional shell elements in Abaqus use bending strain measures that are approximations to those of Koiter-Sanders shell theory (see *About the Abaqus shell element library*). As per the Koiter-Sanders theory the displacement field normal to the shell surface does not produce any bending moments. For example, a purely radial expansion of a cylinder will result in only membrane stresses and strains—there are no variations through the thickness and, hence, no bending. This applies to both the incremental strain measures for linear elastic materials and the deformation gradient for hyperelastic materials. The only exception is for axisymmetric shell elements modeled with hyperelastic materials in Abaqus/Standard. In this case a variation of the membrane stresses and strains through the thickness can occur.

Nodal Mass and Rotary Inertia for Composite Sections

For composite shell sections Abaqus computes the nodal masses based on an average density through the section, weighted with respect to the layer thicknesses. This average density is used to compute an average rotary inertia as if the section were homogeneous. As a consequence, Abaqus does not account for an unsymmetric distribution of mass: the center of mass is assumed to be at the reference surface of the shell. For continuum shells the mass is equally distributed to the top and bottom surface nodes.

Using a Shell Section Integrated during the Analysis to Define the Section Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Shell Elements*
- *Shell Section Behavior*
- **DISTRIBUTION*
- **HOURGLASS STIFFNESS*
- **SHELL SECTION*
- **TRANSVERSE SHEAR STIFFNESS*
- *Creating homogeneous shell sections*
- *Creating composite shell sections*
- *Composite layups*

Overview

A shell section integrated during the analysis:

- is used when numerical integration through the thickness of the shell is required; and
- can be associated with linear or nonlinear material behavior.

Defining a Homogeneous Shell Section

To define a shell made of a single material, use a material definition (*Material Data Definition*) to define the material properties of the section and associate these properties with the section definition. Optionally, you can refer to an orientation (*Orientations*) to be associated with this material definition. A spatially varying local coordinate system defined with a distribution (*Distribution Definition*) can be assigned to the shell section definition. Linear or nonlinear material behavior can be associated with the section definition. However, if the material response is linear, the more economic approach is to use a general shell section (see *Using a General Shell Section to Define the Section Behavior*).

You specify the shell thickness and the number of integration points to be used through the shell section (see below). For continuum shell elements the specified shell thickness is used to estimate certain section properties, such as hourglass stiffness, which are later computed using the actual thickness computed from the element geometry.

You must associate the section properties with a region of your model.

If the orientation definition assigned to a shell section definition is defined with distributions, spatially varying local coordinate systems are applied to all shell elements associated with the shell section. A default local coordinate system (as defined by the distributions) is applied to any shell element that is not specifically included in the associated distribution.

Input File Usage: **SHELL SECTION*, ELSET=*name*, MATERIAL=*name*, ORIENTATION=*name*

where the ELSET parameter refers to a set of shell elements.

Abaqus/CAE Usage: Property module:
Create Section: select **Shell** as the section **Category** and **Homogeneous** as the section **Type**: **Section integration: During analysis**; **Basic**: **Material:** *name*

Assign->Material Orientation: select regions
Assign->Section: select regions

Defining a Composite Shell Section

You can define a laminated (layered) shell made of one or more materials. You specify the thickness, the number of integration points (see below), the material, and the orientation (either as a reference to an orientation definition or as an angle measured relative to the overall orientation definition) for each layer of the shell. The order of the laminated shell layers with respect to the positive direction of the shell normal is defined by the order in which the layers are specified.

Optionally, you can specify an overall orientation definition for the layers of a composite shell. A spatially varying local coordinate system defined with a distribution (*Distribution Definition*) can be used to specify the overall orientation definition for the layers of a composite shell.

For continuum shell elements the thickness is determined from the element geometry and might vary through the model for a given section definition. Hence, the specified thicknesses are only relative thicknesses for each layer. The actual thickness of a layer is the element thickness times the fraction of the total thickness that is accounted for by each layer. The thickness ratios for the layers need not be given in physical units, nor do the sum of the layer relative thicknesses need to add to one. The specified shell thickness is used to estimate certain section properties, such as hourglass stiffness, which are later computed using the actual thickness computed from the element geometry.

Spatially varying thicknesses can be specified on the layers of conventional shell elements using distributions (*Distribution Definition*). A distribution that is used to define layer thickness must have a default value. The default layer thickness is used by any shell element assigned to the shell section that is not specifically assigned a value in the distribution.

An example of a section with three layers and three section points per layer is shown in *Figure 1*.

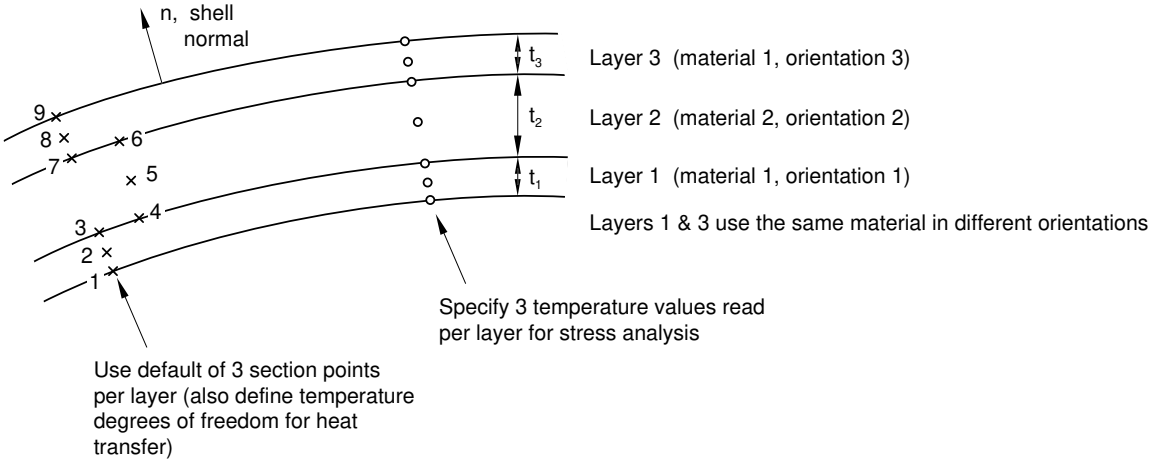


Figure 1: Example of composite shell section definition.

The material name specified for each layer refers to a material definition (*Material Data Definition*). The material behavior can be linear or nonlinear.

The orientation for each layer is specified by either the name of the orientation (*Orientations*) associated with the layer or the orientation angle in degrees for the layer. Spatially varying orientation angles can be specified on a layer using distributions (*Distribution Definition*). Orientation angles, ϕ, are measured positive counterclockwise around the normal and relative to the overall section orientation. If either of the two local directions from the overall section orientation is not in the surface of the shell, ϕ is applied after the section

orientation has been projected onto the shell surface. If you do not specify an overall section orientation, ϕ is measured relative to the default local shell directions (see *Conventions*).

You must associate the section properties with a region of your model.

If the orientation definition assigned to a shell section definition is defined with distributions, spatially varying local coordinate systems are applied to all shell elements associated with the shell section. A default local coordinate system (as defined by the distributions) is applied to any shell element that is not specifically included in the associated distribution.

Unless your model is relatively simple, you will find it increasingly difficult to define your model using composite shell sections as you increase the number of layers and as you assign different sections to different regions. It can also be cumbersome to redefine the sections after you add new layers or remove or reposition existing layers. To manage a large number of layers in a typical composite model, you might want to use the composite layup functionality in Abaqus/CAE. For more information, see *Composite layups*.

Input File Usage: **SHELL SECTION*, ELSET=*name*, COMPOSITE, ORIENTATION=*name*

where the ELSET parameter refers to a set of shell elements.

Abaqus/CAE Usage: Abaqus/CAE uses a composite layup or a composite shell section to define the layers of a composite shell.

Use the following option for a composite layup:

Property module: **Create Composite Layup**: select **Conventional Shell** or **Continuum Shell** as the **Element Type**: **Section integration: During analysis**: specify orientations, regions, and materials

Use the following options for a composite shell section:

Property module:
Create Section: select **Shell** as the section **Category** and **Composite** as the section **Type**: **Section integration: During analysis**
Assign->Material Orientation: select regions
Assign->Section: select regions

Defining the Shell Section Integration

Simpson's rule and Gauss quadrature are provided to calculate the cross-sectional behavior of a shell. You can specify the number of section points through the thickness of each layer and the integration method as described below. The default integration method is Simpson's rule with five points for a homogeneous section and Simpson's rule with three points in each layer for a composite section.

The three-point Simpson's rule and the two-point Gauss quadrature are exact for linear problems. The default number of section points should be sufficient for routine thermal-stress calculations and nonlinear applications (such as predicting the response of an elastic-plastic shell up to limit load). For more severe thermal shock cases or for more complex nonlinear calculations involving strain reversals, more section points might be required; normally no more than nine section points (using Simpson's rule) are required. Gaussian integration normally requires no more than five section points.

Gauss quadrature provides greater accuracy than Simpson's rule when the same number of section points are used. Therefore, to obtain comparable levels of accuracy, Gauss quadrature requires fewer section points than Simpson's rule does and, thus, requires less computational time and storage space.

Using Simpson's Rule

By default, Simpson's rule will be used for the shell section integration. The default number of section points is five for a homogeneous section and three in each layer for a composite section.

Simpson's integration rule should be used if results output on the shell surfaces or transverse shear stress at the interface between two layers of a composite shell is required and must be used for heat transfer and coupled temperature-displacement shell elements.

Input File Usage: **SHELL SECTION*, SECTION INTEGRATION=SIMPSON

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: During analysis**, **Thickness integration rule: Simpson**

Use the following option for a homogeneous or composite shell section:

Property module: shell section editor: **Section integration: During analysis**; **Basic**: **Thickness integration rule: Simpson**

Using Gauss Quadrature

If you use Gauss quadrature for the shell section integration, the default number of section points is three for a homogeneous section and two in each layer for a composite section.

In Gauss quadrature there are no section points on the shell surfaces; therefore, Gauss quadrature should be used only in cases where results on the shell surfaces are not required.

Gauss quadrature cannot be used for heat transfer and coupled temperature-displacement shell elements.

Input File Usage: **SHELL SECTION*, SECTION INTEGRATION=GAUSS

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: During analysis**, **Thickness integration rule: Gauss**

Use the following option for a homogeneous or composite shell section:

Property module: shell section editor: **Section integration: During analysis**; **Basic**: **Thickness integration rule: Gauss**

Defining a Shell Offset Value for Conventional Shells

You can define the distance (measured as a fraction of the shell's thickness) from the shell's midsurface to the reference surface containing the element's nodes (see *Defining the Initial Geometry of Conventional Shell Elements*). Positive values of the offset are in the positive normal direction (see *About Shell Elements*). When the offset is set equal to 0.5, the top surface of the shell is the reference surface. When the offset is set equal to –0.5, the bottom surface is the reference surface. The default offset is 0, which indicates that the middle surface of the shell is the reference surface.

You can specify an offset value that is greater in magnitude than 0.5. However, this technique should be used with caution in regions of high curvature. The element's area and all kinematic quantities are calculated relative

to the reference surface, which might lead to a surface area integration error, affecting the stiffness and mass of the shell.

A spatially varying offset can be defined for conventional shells using a distribution (*Distribution Definition*). The distribution used to define the shell offset must have a default value. The default offset is used by any shell element assigned to the shell section that is not specifically assigned a value in the distribution.

An offset to the shell's top surface is illustrated in *Figure 2*.

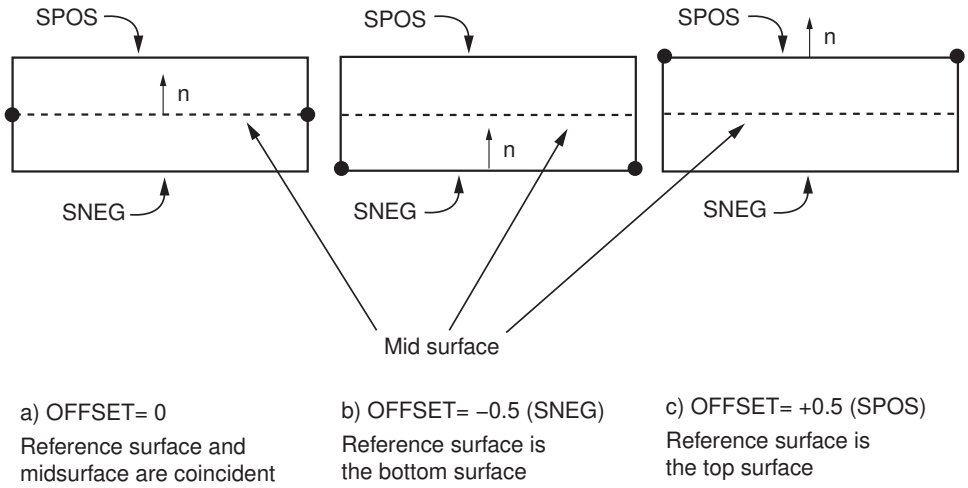


Figure 2: Schematic of shell offset for an offset value of 0.5.

Input File Usage: Use the following option to specify a value for the shell offset:

**SHELL SECTION*, OFFSET=*offset*

The OFFSET parameter accepts a value, a label (SPOS or SNEG), or the name of a distribution that is used to define a spatially varying offset. Specifying SPOS is equivalent to specifying a value of 0.5; specifying SNEG is equivalent to specifying a value of –0.5.

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: During analysis**; **Offset**: choose a reference surface, specify an offset, or select a scalar discrete field

Use the following option for a shell section assignment:

Property module: **Assign->Section**: select regions: **Section**: select a homogeneous or composite shell section: **Definition**: select a reference surface, specify an offset, or select a scalar discrete field

Defining a Variable Thickness for Conventional Shells Using Distributions

You can define a spatially varying thickness for conventional shells using a distribution (*Distribution Definition*). The thickness of continuum shell elements is defined by the element geometry.

For composite shells the total thickness is defined by the distribution, and the layer thicknesses you specify are scaled proportionally such that the sum of the layer thicknesses is equal to the total thickness (including spatially varying layer thicknesses defined with a distribution).

The distribution used to define shell thickness must have a default value. The default thickness is used by any shell element assigned to the shell section that is not specifically assigned a value in the distribution.

If the shell thickness is defined for a shell section with a distribution, nodal thicknesses cannot be used for that section definition.

Input File Usage: Use the following option to define a spatially varying thickness:

**SHELL SECTION*, SHELL THICKNESS=*distribution name*

Abaqus/CAE Usage: Use the following option for a conventional shell composite layup:

Property module: composite layup editor: **Section integration: During analysis**; **Shell Parameters**: **Shell thickness: Element distribution**: select an analytical field or an element-based discrete field

Use the following option for a homogeneous shell section:

Property module: shell section editor: **Section integration: During analysis**; **Basic**: **Shell thickness: Element distribution**: select an analytical field or an element-based discrete field

Use the following option for a composite shell section:

Property module: shell section editor: **Section integration: During analysis**; **Advanced**: **Shell thickness: Element distribution**: select an analytical field or an element-based discrete field

Defining a Variable Nodal Thickness for Conventional Shells

You can define a conventional shell with continuously varying thickness by specifying the thickness of the shell at the nodes. The thickness of continuum shell elements is defined by the element geometry.

If you indicate that the nodal thicknesses will be specified, for homogeneous shells any constant shell thickness you specify will be ignored, and the shell thickness will be interpolated from the nodes. The thickness must be defined at all nodes connected to the element.

For composite shells the total thickness is interpolated from the nodes, and the layer thicknesses you specify are scaled proportionally such that the sum of the layer thicknesses is equal to the total thickness (including spatially varying layer thicknesses defined with a distribution).

If the shell thickness is defined for a shell section with a distribution, nodal thicknesses cannot be used for that section definition. However, if nodal thicknesses are used, you can still use distributions to define spatially varying thicknesses on the layers of conventional shell elements.

Input File Usage: Use both of the following options:

**NODAL THICKNESS*
**SHELL SECTION*, NODAL THICKNESS

Abaqus/CAE Usage: Use the following option for a conventional shell composite layup:

Property module: composite layup editor: **Section integration: During analysis**; **Shell Parameters**: **Nodal distribution**: select an analytical field or a node-based discrete field

Use the following option for a homogeneous shell section:

Property module: shell section editor: **Section integration: During analysis**; **Basic**: **Nodal distribution**: select an analytical field or a node-based discrete field

Use the following option for a composite shell section:

Property module: shell section editor: **Section integration: During analysis**; **Advanced**: **Nodal distribution**: select an analytical field or a node-based discrete field

Defining the Poisson Strain in Shell Elements in the Thickness Direction

Abaqus allows for a possible uniform change in the shell thickness in a geometrically nonlinear analysis (see *Change of Shell Thickness*). The Poisson's strain can be based on a fixed section Poisson's ratio, either user specified or computed by Abaqus based on the elastic portion of the material definition. Alternatively, in Abaqus/Explicit the Poisson strain can be integrated through the section based on the material response at the individual material points in the section.

By default, Abaqus/Standard computes the Poisson's strain using a fixed section Poisson's ratio of 0.5; Abaqus/Explicit uses the material response to compute the Poisson's strain. See *Finite-strain shell element formulation* for details regarding the underlying formulation.

Input File Usage: Use the following option to specify a value for the effective Poisson's ratio:

**SHELL SECTION*, POISSON=ν_{eff}

Use the following option to cause the shell thickness to change based on the element initial elastic material definition:

**SHELL SECTION*, POISSON=ELASTIC

Use the following option (available only in Abaqus/Explicit) to cause the thickness direction strain under plane stress conditions to be a function of the membrane strains and the in-plane material properties:

**SHELL SECTION*, POISSON=MATERIAL

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: During analysis**; **Shell Parameters**: **Section Poisson's ratio: Use analysis default** or **Specify value:** ν_{eff}

Use the following option for a homogeneous or composite shell section:

Property module: shell section editor: **Section integration: During analysis**; **Advanced**: **Section Poisson's ratio: Use analysis default** or **Specify value:** ν_{eff}

You cannot specify a shell thickness direction behavior based on the initial elastic material definition in Abaqus/CAE.

Defining the Thickness Modulus in Continuum Shell Elements

The thickness modulus is used in computing the stress in the thickness direction (see *Thickness Direction Stress in Continuum Shell Elements*). By default, Abaqus computes a thickness modulus that is equal to twice the initial in-plane shear modulus based on the elastic portion of the material definitions in the initial configuration. Alternatively, you can either provide a value (that is, specify it directly) or let Abaqus compute it as the tensile modulus in the out-of-plane direction based on the elastic properties in the initial configuration.

If the material properties are unavailable during the preprocessing stage of input; for example, when the material behavior is defined by the fabric material model or user subroutine *UMAT* or *VUMAT*, you must specify the effective thickness modulus directly.

Input File Usage: Use the following option to define an effective thickness modulus directly:

**SHELL SECTION*, THICKNESS MODULUS=E_{eff}

Use the following option to let Abaqus compute the effective thickness modulus as the initial tensile modulus in the out-of-plane direction:

**SHELL SECTION*, THICKNESS MODULUS=ELASTIC,POISSON=ELASTIC

THICKNESS MODULUS=ELASTIC must be used in conjunction with POISSON=ELASTIC.

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: During analysis**;

Shell Parameters: **Thickness modulus** E_{eff} to specify the thickness properties directly

Use the following option for a homogeneous or composite shell section:

Property module: shell section editor: **Section integration: During analysis**;

Advanced: **Thickness modulus** E_{eff} to specify the thickness properties directly

You cannot specify a shell thickness direction behavior based on the initial elastic material definition in Abaqus/CAE.

Defining the Transverse Shear Stiffness

You can provide nondefault values of the transverse shear stiffness. You *must* specify the transverse shear stiffness in Abaqus if the section is used with shear flexible shells and the material definitions used in the shell section do not include linear elasticity, hypoelasticity, or hyperelasticity. See *Shell Section Behavior* for more information about transverse shear stiffness.

If you do not specify the transverse shear stiffness values, Abaqus integrates through the section to determine them. The transverse shear stiffness is precalculated based on the initial elastic material properties, as defined by the initial temperature and predefined field variables evaluated at the midpoint of each material layer. This stiffness is not recalculated during the analysis.

For most shell sections, including layered composite or sandwich shell sections, Abaqus calculates the transverse shear stiffness values required in the element formulation. You can override these default values. The default shear stiffness values are not calculated in some cases if estimates of the shear moduli are unavailable during the preprocessing stage of input; for example, when the material behavior is defined by the fabric material model or by user subroutines *UMAT*, *UHYPEL*, *UHYPER*, or *VUMAT*. In such cases (except for STRI3 elements), you must specify the material transverse shear modulus (see *Defining the Elastic Transverse Shear Modulus*) based on which Abaqus calculates the transverse shear stiffness values or define the transverse shear stiffness for the shell section directly as described below.

Input File Usage: Use both of the following options:

**SHELL SECTION*
**TRANSVERSE SHEAR STIFFNESS*

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: During analysis**; **Shell Parameters**: toggle on **Specify transverse shear**

Use the following option for a homogeneous or composite shell section:

Property module: shell section editor: **Section integration: During analysis**; **Advanced**: toggle on **Specify transverse shear**

Specifying the Order of Accuracy in the Abaqus/Explicit Shell Element Formulation

In Abaqus/Explicit you can specify second-order accuracy in the shell element formulation. See *Section Controls* for more information.

Input File Usage: **SHELL SECTION*, CONTROLS=*name*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Element Controls**

Defining Density for Conventional Shells

You can define additional mass per unit area for conventional shell elements directly in the section definition. This functionality is similar to the more general functionality of defining a nonstructural mass contribution (see *Nonstructural Mass Definition*.) The only difference between the two definitions is that the nonstructural mass contributes to the rotary inertia terms about the midsurface while the additional mass defined in the section definition does not.

Input File Usage: Use the following option to define the density directly:

**SHELL SECTION*, ELSET=*name*, DENSITY=ρ

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: During analysis**; **Shell Parameters**: toggle on **Density**, and enter ρ

Use the following option for a homogeneous or composite shell section:

Property module: shell section editor: **Section integration: During analysis**; **Advanced**: toggle on **Density**, and enter ρ

Specifying Nondefault Hourglass Control Parameters for Reduced-Integration Shell Elements

You can specify a nondefault hourglass control formulation or scale factors for elements that use reduced integration. See *Section Controls* for more information.

In Abaqus/Standard the nondefault enhanced hourglass control formulation is available only for S4R and SC8R elements. When the enhanced hourglass control formulation is used with composite shells, the average value of the bulk material properties and the minimum value of the shear material properties over all the layers are used for computing the hourglass forces and moments.

In Abaqus/Standard you can modify the default values for hourglass control stiffness based on the default total stiffness approach for elements that use reduced integration and define a scaling factor for the stiffness associated with the drill degree of freedom (rotation about the surface normal) for elements that use six degrees of freedom at a node.

The stiffness associated with the drill degree of freedom is the average of the direct components of the transverse shear stiffness multiplied by a scaling factor. In most cases the default scaling factor is appropriate for constraining the drill rotation to follow the in-plane rotation of the element. If an additional scaling factor is defined, the additional scaling factor should not increase or decrease the drill stiffness by more than a factor of 100.0 for most typical applications. Usually, a scaling factor between 0.1 and 10.0 is appropriate. Continuum shell elements do not use a drill stiffness; hence, the scale factor is ignored.

There are no hourglass stiffness factors or scale factors for hourglass stiffness for the nondefault enhanced hourglass control formulation. You can define the scale factor for the drill stiffness for the nondefault enhanced hourglass control formulation.

Input File Usage: Use both of the following options to specify a nondefault hourglass control formulation or scale factors for reduced-integration elements:

**SECTION CONTROLS*, NAME=*name*
**SHELL SECTION*, CONTROLS=*name*

Use both of the following options in Abaqus/Standard to modify the default values for hourglass control stiffness based on the default total stiffness approach for reduced-integration elements and to define a scaling factor for the stiffness associated with the drill degree of freedom (rotation about the surface normal) for six degree of freedom elements:

**SHELL SECTION*
**HOURGLASS STIFFNESS*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Element Controls**

Specifying Temperature and Field Variables

You can specify temperatures and field variables for conventional shell elements by defining the value at the reference surface of the shell and the gradient through the shell thickness or by defining the values at equally spaced points through each layer of the shell's thickness. You can specify a temperature gradient only for elements without temperature degrees of freedom. The temperatures and field variables for continuum shell elements are defined at the nodes and then interpolated to the section points.

The actual values of the temperatures and field variables are specified as either predefined fields or initial conditions (see *Predefined Fields* or *Initial Conditions*).

If temperature is to be read as a predefined field from the results file or the output database file of a previous analysis, the temperature must be defined at equally spaced points through each layer of the thickness. In addition, the results file must be modified so that the field variable data are stored in record 201. See *Predefined Fields* for additional details.

Defining the Value at the Reference Surface and the Gradient through the Thickness

You can define the temperature or predefined field by its magnitude on the reference surface of the shell and the gradient through the thickness. If only one value is given, the magnitude will be constant through the thickness.

Input File Usage: Use the following option to specify that the temperatures or predefined fields are defined by a gradient:

**SHELL SECTION*

Use any of the following options to specify the actual values of the temperatures or predefined fields:

**TEMPERATURE*
**FIELD*
**INITIAL CONDITIONS*, TYPE=TEMPERATURE
**INITIAL CONDITIONS*, TYPE=FIELD

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: During analysis**; **Shell Parameters**; **Temperature variation: Linear through thickness**

Use the following option for a homogeneous or composite shell section:

Property module: shell section editor: **Section integration: During analysis**: **Advanced**; **Temperature variation: Linear through thickness**

Only initial temperatures and predefined temperature fields are supported in Abaqus/CAE.

Load module: **Create Predefined Field**: **Step:** *initial_step* or *analysis_step*: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**

Defining the Values at Equally Spaced Points through the Thickness

Alternatively, you can define the temperature and field variable values at equally spaced points through the thickness of a shell or of each layer of a composite shell.

For a sequentially coupled thermal-stress analysis in Abaqus/Standard, the number (n) of equally spaced points through the thickness of a layer is an odd number when temperature values are obtained from the results file or the output database file generated by a previous Abaqus/Standard heat transfer analysis (since only Simpson's rule can be used for integration through the section in heat transfer analysis). n may be even or odd if the values are supplied from some other source. In either case Abaqus/Standard interpolates linearly between the two closest defined temperature points to find the temperature values at the section points.

The number of predefined field points through each layer, *n*, must be the same as the number of integration points used through the same layer in the analysis from which the temperatures are obtained. This requirement implies that in the previous analysis each of the layers must have the same number of integration points.

You specify $1 + n_l\,(n_T - 1)$ temperature or field variable values, where n_l is the number of layers in the shell section and n_T ($n_T > 1$) is the value of *n*. For n_T=1, you specify n_l temperature or field variable value for a given node or node set.

Input File Usage: Use the following option to specify that the temperatures or predefined fields are defined at equally spaced points:

**SHELL SECTION*, TEMPERATURE=*n*

Use any of the following options to specify the actual values of the temperatures or predefined fields:

**TEMPERATURE*
**FIELD*
**INITIAL CONDITIONS*, TYPE=TEMPERATURE
**INITIAL CONDITIONS*, TYPE=FIELD

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: During analysis**; **Shell Parameters**; **Temperature variation: Piecewise linear over *n* values**

Use the following option for a homogeneous or composite shell section:

Property module: shell section editor: **Section integration: During analysis**: **Advanced**; **Temperature variation: Piecewise linear over *n* values**

Only initial temperatures and predefined temperature fields are supported in Abaqus/CAE.

Load module: **Create Predefined Field**: **Step:** *initial_step* or *analysis_step*: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**

Example

An example of this scheme is illustrated in *Figure 3* and *Figure 4*.

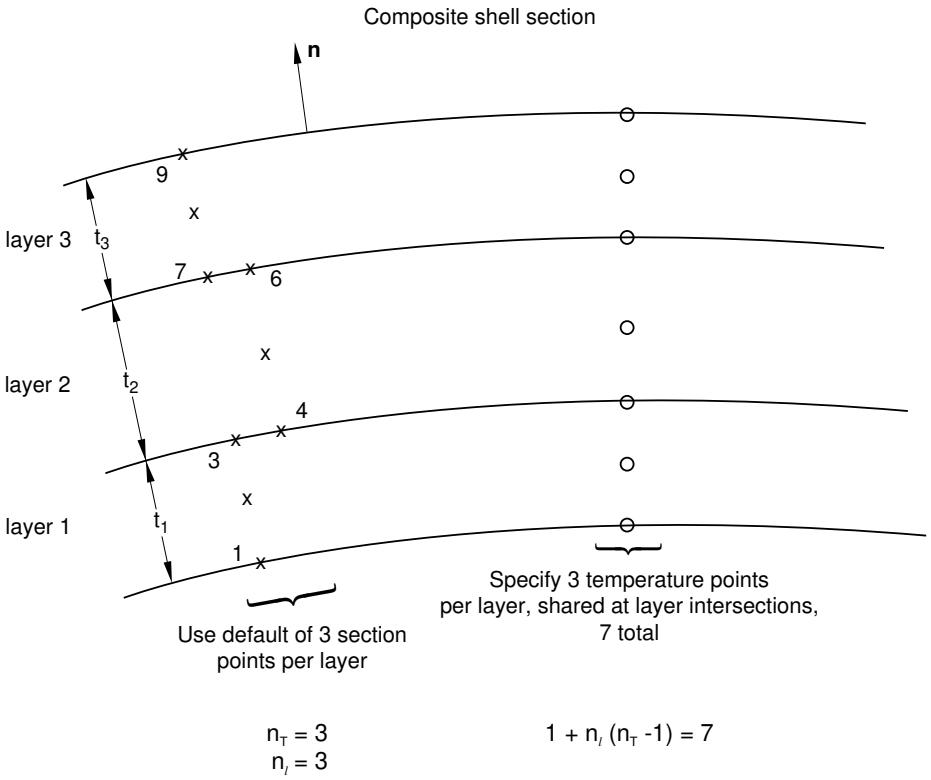


Figure 3: Defining temperature values at n equally spaced points using Simpson's rule.

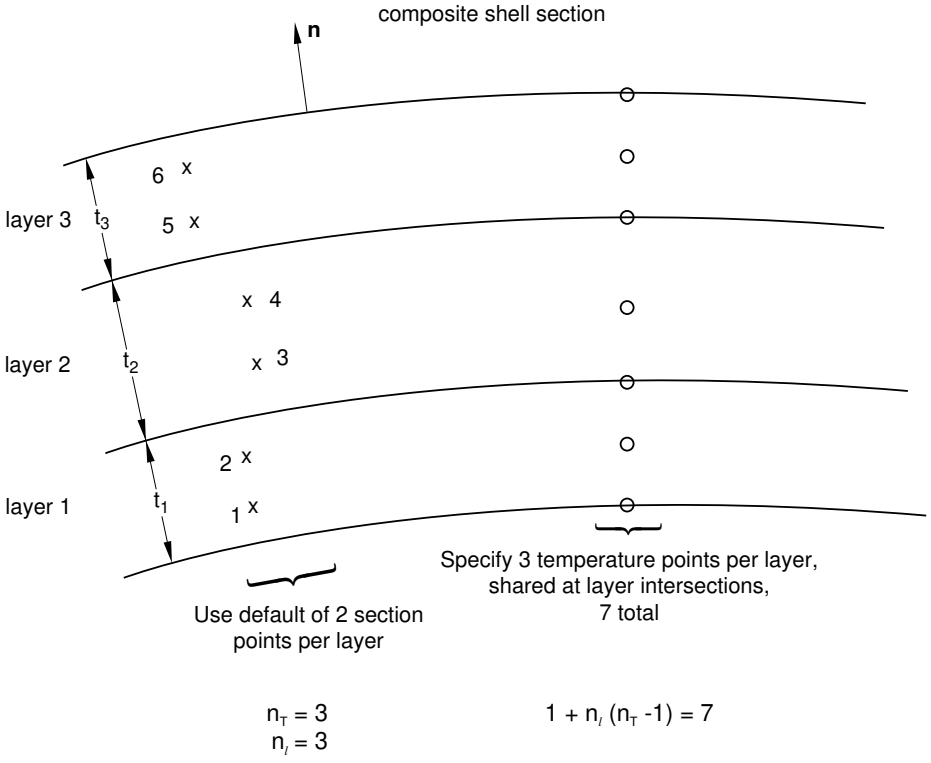


Figure 4: Defining temperature values at n equally spaced points using Gauss integration.

The following Abaqus/Standard heat transfer shell section definition corresponds to this example:

```
*SHELL SECTION, COMPOSITE
t1, 3, MAT1, ORI1
t2, 3, MAT2, ORI2
t3, 3, MAT3, ORI3
```

This creates degrees of freedom 11–17 in the heat transfer analysis. Temperatures corresponding to these degrees of freedom are then read into the stress analysis at the temperature points shown and interpolated to the section points shown.

Defining a Continuous Temperature Field

In Abaqus/Standard if an element with temperature degrees of freedom other than a shell abuts the bottom surface of a shell element with temperature degrees of freedom, the temperature field is made continuous when the elements share nodes. If another element with temperature degrees of freedom abuts the top surface, separate nodes must be used and a linear constraint equation (*Linear Constraint Equations*) must be used to constrain the temperatures to be the same (that is, to give the same value to the top surface degree of freedom on the shell and degree of freedom 11 on the other element).

For the same reason you must be careful if a different number of temperature points is used in adjacent shell elements. For compatibility MPCs (*General Multi-Point Constraints*) or equation constraints are also needed in this case.

In Abaqus/Explicit since no thermal MPCs and no thermal equation constraints are available for degrees of freedom greater than 11, care must be taken when using a different number of temperature points in adjacent shell elements. This should usually have a localized effect on the temperature distribution, but it might affect the overall solution for the cases in which the temperature gradient through the thickness is significant.

In both Abaqus/Standard and Abaqus/Explicit be careful in the models in which the shell's normals are reversed. In this case the temperature at the bottom of the shell becomes the temperature at the top of the adjacent shell. This might have a small impact on the overall solution for the cases in which the thermal gradient through the thickness is negligible and the temperature variation is mainly in plane. However, if the temperature gradient through the thickness is significant, it might lead to incorrect results.

Output

In an Abaqus/Standard stress analysis, temperature output at the section points can be obtained using the element variable TEMP.

If the temperature values were specified at equally spaced points through the thickness, output at the temperature points can be obtained in an Abaqus/Standard stress analysis, as in a heat transfer analysis, by using the nodal variable NT*xx*. This nodal output variable is also available in Abaqus/Explicit for coupled temperature-displacement analyses. The nodal variable NT*xx* should not be used for output at the temperature points with the default gradient method. In this case output variable NT should be requested; NT11 (the reference temperature value) and NT12 (the temperature gradient) will be output automatically. For continuum shell elements, there is only NT11; all other NT*xx* are irrelevant.

Other output variables that are relevant for shells are listed in each of the library sections describing the specific shell elements. For example, stresses, strains, section forces and moments, average section stresses, section strains, etc. can be output. The section moments are calculated relative to the reference surface.

Using a General Shell Section to Define the Section Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Shell Elements*
- *Shell Section Behavior*
- *UGENS*
- **DISTRIBUTION*
- **HOURGLASS STIFFNESS*
- **SHELL GENERAL SECTION*
- **TRANSVERSE SHEAR STIFFNESS*
- *Creating homogeneous shell sections*
- *Creating composite shell sections*
- *Creating general shell stiffness sections*
- *Composite layups*

Overview

A general shell section:

- is used when numerical integration through the thickness of the shell is not required;
- can be associated with linear elastic material behavior;
- can invoke user subroutine *UGENS* (Abaqus/Standard) or *VUGENS* (Abaqus/Explicit) to define nonlinear section properties in terms of forces and moments;
- can be used to model an equivalent shell section for some more complex geometry (for example, replacing a corrugated shell with an equivalent smooth plate for global analysis); and
- cannot be used with heat transfer and coupled temperature-displacement shells.

Defining the Shell Section Behavior

A general shell section can be defined as follows:

- The section response can be specified by associating the section with a material definition or, in the case of a composite shell, with several different material definitions.
- The section properties can be specified directly.
- In Abaqus/Standard the section response can be programmed in user subroutine *UGENS*.

Specifying the Equivalent Section Properties by Defining the Layers (Thickness, Material, and Orientation)

You can define the shell section's mechanical response by specifying the thickness; the material reference; and the orientation of the section or, for a composite shell, the orientation of each of its layers. Abaqus will determine the equivalent section properties. You must associate the section behavior with a region of your model.

The linear elastic material behavior is defined with a material definition (*Material Data Definition*), which may contain linear elastic behavior (*Linear Elastic Behavior*) and thermal expansion behavior (*Thermal Expansion*). The density (*Density*) and damping (*Material Damping*) behavior can also be specified as described below; in Abaqus/Explicit the density of the material must be defined. However, no nonlinear material properties, such as

plastic behavior, can be included since Abaqus will precompute the section response and will not update that response during the analysis. Dependence of the linear elastic material behavior on temperature or predefined field variables is not allowed.

The shell section response is defined by

$$\{\mathbf{N}\} = [\mathbf{D}] : \{\mathbf{E}\} - \{\mathbf{N^{th}}\}.$$

No temperature-dependent scaling of the modulus is included. The section forces and moments caused by thermal strains, $\{\mathbf{N^{th}}\}$, vary linearly with temperature and are defined by

$$\{\mathbf{N^{th}}\} = (\theta - \theta^I)\ \{\overline{\mathbf{F}}\},$$

where $\{\overline{\mathbf{F}}\}$ are the generalized stresses caused by a fully constrained unit temperature rise that result from the user-defined thermal expansion, θ is the temperature, and θ^I is the initial (stress-free) temperature at this point in the shell (defined by the initial nodal temperatures given as initial conditions; see *Defining Initial Temperatures*).

Defining a Shell Made of a Single Linear Elastic Material

To define a shell made of a single linear elastic material, you refer to the name of a material definition (*Material Data Definition*) as described above. Optionally, you can define an orientation definition to be used with the section (*Orientations*). A spatially varying local coordinate system defined with a distribution (*Distribution Definition*) can be assigned to the shell section definition. In addition, you specify the shell thickness as part of the section definition. For continuum shell elements the specified thickness is used to estimate certain section properties, such as hourglass stiffness, that are later computed from the element geometry.

You must associate this section behavior with a region of your model.

You can redefine the thickness, offset, section stiffness, and material orientation specified in the section definition on an element-by-element basis. See *Distribution Definition*.

If the orientation definition assigned to a shell section definition is defined with distributions, spatially varying local coordinate systems are applied to all shell elements associated with the shell section. A default local coordinate system (as defined by the distributions) is applied to any shell element that is not specifically included in the associated distribution.

Input File Usage: **SHELL GENERAL SECTION*, ELSET=*name*, MATERIAL=*name*, ORIENTATION=*name*

where the ELSET parameter refers to a set of shell elements.

Abaqus/CAE Usage: Property module:
Create Section: select **Shell** as the section **Category** and **Homogeneous** as the section **Type**: **Section integration: Before analysis**; **Basic**: **Material:** *name*
Assign->Material Orientation: select regions
Assign->Section: select regions

Defining a Shell Made of Layers with Different Linear Elastic Material Behaviors

You can define a shell made of layers with different linear elastic material behaviors. Optionally, you can define an orientation definition to be used with the section (*Orientations*). A spatially varying local coordinate system defined with a distribution (*Distribution Definition*) can be assigned to the shell section definition.

You specify the layer thickness; the name of the material forming this layer (as described above); and the orientation angle, ϕ, (in degrees) measured positive counterclockwise relative to the specified section orientation definition. Spatially varying orientation angles can be specified on a layer using distributions (*Distribution*

Definition). If either of the two local directions from the specified section orientation is not in the surface of the shell, ϕ is applied after the section orientation has been projected onto the shell surface. If you do not specify a section orientation, ϕ is measured relative to the default shell local directions (see *Conventions*). The order of the laminated shell layers with respect to the positive direction of the shell normal is defined by the order in which the layers are specified.

For continuum shell elements the thickness is determined from the element geometry and might vary through the model for a given section definition. Hence, the specified thicknesses are only relative thicknesses for each layer. The actual thickness of a layer is the element thickness times the fraction of the total thickness that is accounted for by each layer. The thickness ratios for the layers need not be given in physical units, nor do the sum of the layer relative thicknesses need to add to one. The specified shell thickness is used to estimate certain section properties, such as hourglass stiffness, that are later computed from the element geometry.

Spatially varying thicknesses can be specified on the layers of conventional shell elements (not continuum shell elements) using distributions (*Distribution Definition*). A distribution that is used to define layer thickness must have a default value. The default layer thickness is used by any shell element assigned to the shell section that is not specifically assigned a value in the distribution.

You must associate this section behavior with a region of your model.

If you define the orientation definition assigned to a shell section definition with distributions, spatially varying local coordinate systems are applied to all shell elements associated with the shell section. A default local coordinate system (as defined by the distributions) is applied to any shell element that is not specifically included in the associated distribution.

If you define spatially varying orientation angles or spatially varying thickness with distributions, default data in the distribution table are used to generate the model section summary information (stiffness, thickness, and orientation angle), which are written to the data (`.dat`) file when requested.

Unless your model is relatively simple, you will find it increasingly difficult to define your model using composite shell sections as you increase the number of layers and as you assign different sections to different regions. It can also be cumbersome to redefine the sections after you add new layers or remove or reposition existing layers. To manage a large number of layers in a typical composite model, you might want to use the composite layup functionality in Abaqus/CAE. For more information, see *Composite layups*.

Input File Usage: **SHELL GENERAL SECTION*, ELSET=*name*, COMPOSITE, ORIENTATION=*name*

where the ELSET parameter refers to a set of shell elements.

Abaqus/CAE Usage: Abaqus/CAE uses a composite layup or a composite shell section to define a shell made of layers with different linear elastic material behaviors.

Use the following option for a composite layup:

Property module: **Create Composite Layup**: select **Conventional Shell** or **Continuum Shell** as the **Element Type**: **Section integration: Before analysis**: specify orientations, regions, and materials

Use the following options for a composite shell section:

Property module:
Create Section: select **Shell** as the section **Category** and **Composite** as the section **Type**: **Section integration: Before analysis**
Assign->Material Orientation: select regions
Assign->Section: select regions

Specifying the Equivalent Section Properties Directly for Conventional Shells

You can define the section's mechanical response by specifying the general section stiffness and thermal expansion response—$[\mathbf{D}]$, $\{\mathbf{F}\}$, $Y(\theta, f_\beta)$, and $\alpha(\theta, f_\beta)$, as defined below—directly. Since this method then provides the complete specification of the section's mechanical response, no material reference is needed. Optionally, you can define θ^0, the reference temperature for thermal expansion.

You must associate this section behavior with a region of your model.

In this case the shell section response is defined by

$$\{\mathbf{N}\} = Y(\theta, f_\beta)\ [\mathbf{D}] : \{\mathbf{E}\} - \left\{\mathbf{N^{th}}\right\},$$

where

$\{\mathbf{N}\}$
: are the forces and moments on the shell section (membrane forces per unit length, bending moments per unit length);

$\{\mathbf{E}\}$
: are the generalized section strains in the shell (reference surface strains and curvatures);

$[\mathbf{D}]$
: is the section stiffness matrix;

$Y(\theta, f_\beta)$
: is a scaling modulus, which can be used to introduce temperature (θ) and field-variable (f_β) dependence of the cross-section stiffness; and

$\left\{\mathbf{N^{th}}\right\}$
: are the section forces and moments (per unit length) caused by thermal strains.

These thermal forces and moments in the shell are generated according to the formula

$$\left\{\mathbf{N^{th}}\right\} = \left(\alpha(\theta, f_\beta)\ \left(\theta - \theta^0\right) - \alpha\left(\theta^I, f_\beta^I\right)\ \left(\theta^I - \theta^0\right)\right)\ \{\mathbf{F}\},$$

where

$\alpha(\theta, f_\beta)$
: is a scaling factor (the “thermal expansion coefficient”);

θ^I
: is the initial (stress-free) temperature at this point in the shell, defined by the initial nodal temperatures given as initial conditions (*Defining Initial Temperatures*); and

$\{\mathbf{F}\}$
: are the user-specified generalized section forces and moments (per unit length) caused by a fully constrained unit temperature rise.

If the coefficient of thermal expansion, α, is not a function of temperature, the value of θ^0 is not needed. Note the distinction between θ^0, the reference value used in defining α, and the stress-free initial temperature, θ^I.

In these equations the order of the terms is

$$\{\mathbf{N}\} = \begin{Bmatrix} N_{11} \\ N_{22} \\ N_{12} \\ M_{11} \\ M_{22} \\ M_{12} \end{Bmatrix}, \quad \{\mathbf{E}\} = \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \gamma_{12} \\ \kappa_{11} \\ \kappa_{22} \\ \kappa_{12} \end{Bmatrix};$$

that is, the direct membrane terms come first, then the shear membrane term, then the direct and shear bending terms, with six terms in all. Engineering measures of shear membrane strain (γ_{12}) and twist (κ_{12}) are used in Abaqus.

This method of defining the shell section properties cannot be used with variable thickness shells or continuum shell elements.

See *Laminated composite shells: buckling of a cylindrical panel with a circular hole* for more information.

The stiffness matrix, $[\mathbf{D}]$, can be defined as a constant stiffness for the section or as a spatially varying stiffness by referring to a distribution (*Distribution Definition*). If a spatially varying stiffness is used, the distribution must have a default stiffness defined. The default stiffness is used by any shell element assigned to the shell section that is not specifically assigned a value in the distribution.

Input File Usage: **SHELL GENERAL SECTION*, ELSET=*name*, ZERO=θ^0

where the ELSET parameter refers to a set of shell elements.

Abaqus/CAE Usage: Property module:
Create Section: select **Shell** as the section **Category** and **General shell stiffness** as the section **Type**
Assign->Section: select regions

Specifying the Section Properties in User Subroutine UGENS or VUGENS

You can define the section response in user subroutine *UGENS* (Abaqus/Standard) or *VUGENS* (Abaqus/Explicit) for the more general case where the section response might be nonlinear. User subroutines *UGENS* and *VUGENS* are particularly useful if the nonlinear behavior of the section involves geometric as well as material nonlinearity, such as might occur due to section collapse. If only nonlinear material behavior is present, it is simpler to use a shell section integrated during the analysis with the appropriate nonlinear material model.

You must specify a constant section thickness as part of the section definition or a continuously varying thickness by defining the thickness at the nodes as described below. Even though the section's mechanical behavior is defined in user subroutines *UGENS* and *VUGENS*, the thickness of the shell section is required for calculation of the hourglass control stiffness. You must associate this section behavior with a region of your model.

Abaqus/Standard calls user subroutine *UGENS* for each integration point at each iteration of every increment. The subroutine provides the following information:

- Section state at the start of the increment (section forces and moments, $\mathbf{N}$; generalized section strains, $\mathbf{E}$; solution-dependent state variables; temperature; and any predefined field variables).
- Increments in temperature and predefined field variables.

- Generalized section strain increments, $\Delta\mathbf{E}$.
- Time increment.

Abaqus/Explicit calls user subroutine *VUGENS* for blocks of integration points in shell elements at every increment. The subroutine provides the following information:

- Section states at the start of the increment (section forces, $\boldsymbol{N}$, and moments, $\boldsymbol{M}$; midsurface deformation gradient, $\boldsymbol{F}$; solution-dependent state variables; temperature; and any predefined field variables).
- Section states at the end of the increment (midsurface deformation gradient, $\boldsymbol{F}$; curvature, $\boldsymbol{\kappa}$; temperature; and any predefined field variables).
- Membrane strain increments, $\Delta\boldsymbol{\varepsilon}$.
- Incremental curvature, $\Delta\boldsymbol{\kappa}$.
- Time increment.

Each subroutine must perform two functions: it must update the forces, the moments, and the solution-dependent state variables to their values at the end of the increment; and it must provide the section stiffness matrix, $\partial\mathbf{N}/\partial\mathbf{E}$. In Abaqus/Explicit the section stiffness is used to evaluate the stable time increment. You must program the complete section response, including the thermal expansion effects, in the user subroutine.

In Abaqus/Standard you should ensure that the strain increment is not used or changed in user subroutine *UGENS* for linear perturbation analyses. For this case the quantity is undefined.

You cannot use this method of defining the shell section properties with continuum shell elements.

Input File Usage: **SHELL GENERAL SECTION*, ELSET=*name*, USER

where the ELSET parameter refers to a set of shell elements.

Abaqus/CAE Usage: User subroutines *UGENS* and *VUGENS* are not supported in Abaqus/CAE.

Defining Whether or Not the Section Stiffness Matrices Are Symmetric in Abaqus/Standard

If the section stiffness matrices are not symmetric, you can specify that Abaqus/Standard should use its unsymmetric equation solution capability (see *Defining an Analysis*).

Input File Usage: **SHELL GENERAL SECTION*, ELSET=*name*, USER, UNSYMM

Abaqus/CAE Usage: User subroutine *UGENS* is not supported in Abaqus/CAE.

Defining the Section Properties

Any number of constants can be defined to be used in determining the section behavior. You can specify the number of integer property values required, *m*, and the number of real (floating point) property values required, *n*; the total number of values required is the sum of these two numbers. The default number of integer property values required is 0, and the default number of real property values required is 0.

Integer property values can be used inside user subroutines *UGENS* and *VUGENS* as flags, indices, counters, etc. Examples of real (floating point) property values are material properties, geometric data, and any other information required to calculate the section response in *UGENS* and *VUGENS*.

The property values are passed into user subroutines *UGENS* and *VUGENS* each time the subroutines are called.

Input File Usage: **SHELL GENERAL SECTION*, ELSET=*name*, USER, I PROPERTIES=*m*, PROPERTIES=*n*

To define the property values, enter all floating point values on the data lines first, followed immediately by the integer values. You can enter eight values per line.

Abaqus/CAE Usage: User subroutines *UGENS* and *VUGENS* are not supported in Abaqus/CAE.

Defining the Number of Solution-Dependent Variables That Must Be Stored for the Section

You can define the number of solution-dependent state variables that must be stored at each integration point within the section. There is no restriction on the number of variables associated with a user-defined section. The default number of variables is 1. Examples of such variables are plastic strains, damage variables, failure indices, and user-defined output quantities.

These solution-dependent state variables can be calculated and updated in user subroutines *UGENS* and *VUGENS*.

Input File Usage: **SHELL GENERAL SECTION*, ELSET=*name*, USER, VARIABLES=*n*

Abaqus/CAE Usage: User subroutines *UGENS* and *VUGENS* are not supported in Abaqus/CAE.

Defining Element Deletion and Damage of Transverse Shear Stiffness in Abaqus/Explicit

You can control element deletion and damage of the transverse shear stiffness by defining the solution-dependent state variables to be stored at each integration point within the section. These solution-dependent state variables can be updated inside user subroutine *VUGENS*.

You can control element deletion in a mesh during an Abaqus analysis using user subroutine *VUGENS* to set the state variable DELETE. Deleted elements have no ability to carry stresses and do not contribute to the model stiffness. You specify the state variable number controlling the element deletion flag. For example, specifying a state variable number of n_1 indicates that the state variable SDVn_1 is the deletion flag in the user subroutine. You can set the state variable DELETE to a value of one to indicate that the element is active or to zero to indicate that Abaqus should delete the element from the model.

You can control the transverse shear stiffness using user subroutine *VUGENS* to set the state variable TVS DAMAGE. You specify the state variable number controlling the shell element transverse shear damage variable. For example, specifying a state variable number of n_2 indicates that the state variable SDVn_2 is the transverse shear damage variable in the user subroutine. You can set the state variable TVS DAMAGE to a value between zero and one, with a default value of one indicating the initial undamaged state. This state variable is used as the transverse shear stiffness scaling factor to scale the transverse shear stiffness of the shell elements during an Abaqus analysis.

Input File Usage: **SHELL GENERAL SECTION*, ELSET=*name*, USER, DELETE=n_1, TVS DAMAGE=n_2

Abaqus/CAE Usage: User subroutine *VUGENS* is not supported in Abaqus/CAE.

Idealizing the Section Response

Idealizations allow you to modify the stiffness coefficients in a shell section based on assumptions about the shell's makeup or expected behavior. The following idealizations are available for general shell sections:

- Retain only the membrane stiffness for shells whose predominant response will be in-plane stretching.
- Retain only the bending stiffness for shells whose predominant response will be pure bending.
- Ignore the effects of the material layer stacking sequence for composite shells.

The membrane stiffness and bending stiffness idealizations can be applied to homogeneous shell sections, composite shell sections, or shell sections with the stiffness coefficients specified directly. The idealization to ignore stacking effects can be applied only to composite shell sections.

Idealizations modify the shell general stiffness coefficients after they have been computed normally, including the effects of offset.

- If you use any idealization, all membrane-bending coupling terms are set to zero.
- If you retain only the membrane stiffness, off-diagonal terms of the bending submatrix are set to zero, and diagonal bending terms are set to 1×10^{-6} times the largest diagonal membrane coefficient.
- If you retain only the bending stiffness, off-diagonal terms of the membrane submatrix are set to zero, and diagonal membrane terms are set to 1×10^{-6} times the largest diagonal bending coefficient.
- If you ignore the material layer stacking sequence in a composite shell, each term of the bending submatrix is set equal to $T^2/12$ times the corresponding membrane submatrix term, where T is the total thickness of the shell.

Input File Usage: Use the following option to retain only the membrane stiffness:

**SHELL GENERAL SECTION*, MEMBRANE ONLY

Use the following option to retain only the bending stiffness:

**SHELL GENERAL SECTION*, BENDING ONLY

Use the following option to ignore the effects of the layer stacking sequence:

**SHELL GENERAL SECTION*, COMPOSITE, SMEAR ALL LAYERS

Multiple idealization options can be used on the same general shell section.

Abaqus/CAE Usage: Use any of the following options to apply an idealization to a shell section:

Property module: Homogeneous shell section editor: **Section integration: Before analysis**; **Basic**: **Idealization: Membrane only** or **Bending only**

Property module: Composite shell section editor: **Section integration: Before analysis**; **Basic**: **Idealization: Membrane only**, **Bending only**, or **Smear all layers**

Property module: Shell (conventional or continuum) composite layup editor: **Section integration: Before analysis**; **Basic**: **Idealization: Membrane only**, **Bending only**, or **Smear all layers**

You cannot apply multiple idealizations to the same shell section in Abaqus/CAE, and you cannot apply idealizations to a general shell stiffness section.

Defining a Shell Offset Value for Conventional Shells

You can define the distance (measured as a fraction of the shell's thickness) from the shell's midsurface to the reference surface containing the element's nodes (see *Defining the Initial Geometry of Conventional Shell Elements*). Positive values of the offset are in the positive normal direction (see *About Shell Elements*). When the offset is set equal to 0.5, the top surface of the shell is the reference surface. When the offset is set equal to –0.5, the bottom surface is the reference surface. The default offset is 0, which indicates that the middle surface of the shell is the reference surface.

You can specify an offset value that is greater in magnitude than 0.5. However, this technique should be used with caution in regions of high curvature. The element's area and all kinematic quantities are calculated relative to the reference surface, which might lead to a surface area integration error, affecting the stiffness and mass of the shell.

A spatially varying offset can be defined for conventional shells using a distribution (*Distribution Definition*). The distribution used to define the shell offset must have a default value. The default offset is used by any shell element assigned to the shell section that is not specifically assigned a value in the distribution.

An offset to the shell's top surface is illustrated in *Figure 1*.

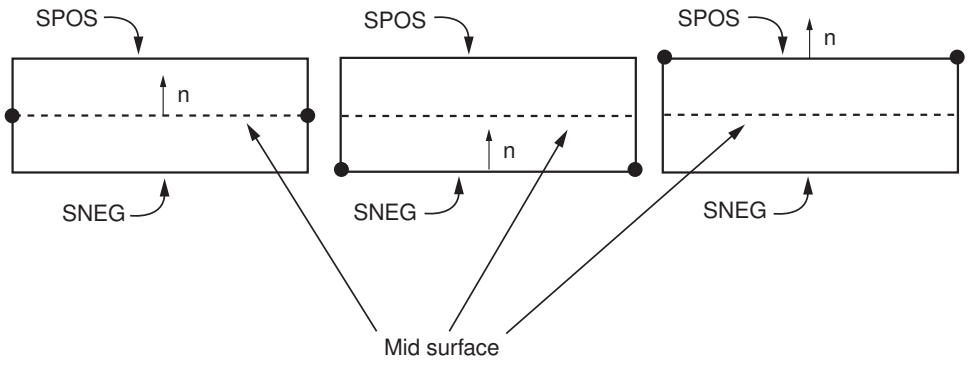


Figure 1: Schematic of shell offset for an offset value of 0.5.

A shell offset value can be specified only if a material definition is referenced or a composite shell section is defined.

Input File Usage: Use the following option to specify a value for the shell offset:

**SHELL GENERAL SECTION*, OFFSET=*offset*

The OFFSET parameter accepts a value, a label (SPOS or SNEG), or the name of a distribution that is used to define a spatially varying offset. Specifying SPOS is equivalent to specifying a value of 0.5; specifying SNEG is equivalent to specifying a value of –0.5.

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: Before analysis**; **Offset**: choose a reference surface, specify an offset, or select a scalar discrete field

Use the following option for a shell section assignment:

Property module: **Assign**->**Section**: select regions: **Section**: select a homogeneous or composite shell section: **Definition**: select a reference surface, specify an offset, or select a scalar discrete field

Defining a Variable Thickness for Conventional Shells Using Distributions

You can define a spatially varying thickness for conventional shells using a distribution (*Distribution Definition*). The thickness of continuum shell elements is defined by the element geometry.

For composite shells the total thickness is defined by the distribution. The layer thicknesses that you specify are scaled proportionally such that the sum of the layer thicknesses is equal to the total thickness (including spatially varying layer thicknesses defined with a distribution).

The distribution used to define shell thickness must have a default value. The default thickness is used by any shell element assigned to the shell section that is not specifically assigned a value in the distribution.

If you define spatially varying thickness with a distribution, default data in the distribution table are used to generate the model section summary information (stiffness and thickness), which are written to the data (`.dat`) file when requested.

If the shell thickness is defined for a shell section with a distribution, nodal thicknesses cannot be used for that section definition.

Input File Usage: Use the following option to define a spatially varying thickness:

**SHELL GENERAL SECTION*, SHELL THICKNESS=*distribution name*

Abaqus/CAE Usage: Use the following option for a conventional shell composite layup:

Property module: composite layup editor: **Section integration: Before analysis**; **Shell Parameters**: **Shell thickness: Element distribution**: select an analytical field or an element-based discrete field

Use the following option for a homogeneous shell section:

Property module: shell section editor: **Section integration: Before analysis**; **Basic**: **Shell thickness: Element distribution**: select an analytical field or an element-based discrete field

Use the following option for a composite shell section:

Property module: shell section editor: **Section integration: Before analysis**; **Advanced**: **Shell thickness: Element distribution**: select an analytical field or an element-based discrete field

Defining a Variable Nodal Thickness for Conventional Shells

You can define a conventional shell with continuously varying thickness by specifying the thickness of the shell at the nodes. This method can be used only if the section is defined in terms of material properties; it cannot be used if the section behavior is defined by specifying the equivalent section properties directly. For continuum shell elements a continuously varying thickness can be defined through the element nodal geometry; hence, the nodal thickness is not meaningful.

If you indicate that the nodal thicknesses will be specified, for homogeneous shells any constant shell thickness you specify will be ignored, and the shell thickness will be interpolated from the nodes. The thickness must be defined at all nodes connected to the element.

For composite shells the total thickness is interpolated from the nodes, and the layer thicknesses you specify are scaled proportionally such that the sum of the layer thicknesses is equal to the total thickness (including spatially varying layer thicknesses defined with a distribution).

If the shell thickness is defined for a shell section with a distribution, nodal thicknesses cannot be used for that section definition. However, if nodal thicknesses are used, you can still use distributions to define spatially varying thicknesses on the layers of conventional shell elements.

Input File Usage: Use both of the following options:

**NODAL THICKNESS*
**SHELL GENERAL SECTION*, NODAL THICKNESS

Abaqus/CAE Usage: Use the following option for a conventional shell composite layup:

Property module: composite layup editor: **Section integration: Before analysis**; **Shell Parameters**: **Nodal distribution**: select an analytical field or a node-based discrete field

Use the following option for a homogeneous shell section:

Property module: shell section editor: **Section integration: Before analysis**; **Basic**: **Nodal distribution**: select an analytical field or a node-based discrete field

Use the following option for a composite shell section:

Property module: shell section editor: **Section integration: Before analysis**; **Advanced**: **Nodal distribution**: select an analytical field or a node-based discrete field

Defining the Poisson Strain in Shell Elements in the Thickness Direction

Abaqus allows for a possible uniform change in the shell thickness in a geometrically nonlinear analysis (see *Change of Shell Thickness*). The Poisson's strain is based on a fixed section Poisson's ratio, either user specified or computed by Abaqus based on the elastic portion of the material definition.

By default, Abaqus computes the Poisson's strain using a fixed section Poisson's ratio of 0.5.

Input File Usage: Use the following option to specify a value for the effective Poisson's ratio:

**SHELL GENERAL SECTION*, POISSON=ν_{eff}

Use the following option to cause the shell thickness to change based on the initial elastic properties of the material:

**SHELL GENERAL SECTION*, POISSON=ELASTIC

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: Before analysis**; **Shell Parameters**: **Section Poisson's ratio: Use analysis default** or **Specify value:** ν_{eff}

Use the following option for a homogeneous or composite shell section:

Property module: shell section editor: **Section integration: Before analysis**; **Advanced**: **Section Poisson's ratio: Use analysis default** or **Specify value:** ν_{eff}

You cannot specify a shell thickness direction behavior based on the initial elastic material definition in Abaqus/CAE.

Defining the Thickness Modulus in Continuum Shell Elements

The thickness modulus is used in computing the stress in the thickness direction (see *Thickness Direction Stress in Continuum Shell Elements*). By default, Abaqus computes a thickness modulus that is equal to twice the initial in-plane shear modulus based on the elastic portion of the material definitions in the initial configuration. Alternatively, you can either provide a value (that is, specify it directly) or let Abaqus compute it as the tensile modulus in the out-of-plane direction based on the elastic properties in the initial configuration.

If the material properties are unavailable during the preprocessing stage of input (for example, when the material behavior is defined by the fabric material model or user subroutine *UMAT* or *VUMAT*), you must specify the effective thickness modulus directly.

Input File Usage: Use the following option to define an effective thickness modulus directly:

**SHELL GENERAL SECTION*, THICKNESS MODULUS=E_{eff}

Use the following option to let Abaqus compute the effective thickness modulus as the initial tensile modulus in the out-of-plane direction:

**SHELL SECTION*, THICKNESS MODULUS=ELASTIC,POISSON=ELASTIC

THICKNESS MODULUS=ELASTIC must be used in conjunction with POISSON=ELASTIC.

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: Before analysis**; **Shell Parameters**: **Thickness modulus** E_{eff} to specify the thickness properties directly

Use the following option for a homogeneous or composite shell section:

Property module: shell section editor: **Section integration: Before analysis**; **Advanced**: **Thickness modulus** E_{eff} to specify the thickness properties directly

Defining the Transverse Shear Stiffness

You can provide nondefault values of the transverse shear stiffness. You must specify the transverse shear stiffness for shear flexible shells in Abaqus/Standard if the section properties are specified in user subroutine

UGENS. If you do not specify the transverse shear stiffness, it will be calculated as described in *Shell Section Behavior*.

Input File Usage: Use both of the following options:

**SHELL GENERAL SECTION*
**TRANSVERSE SHEAR STIFFNESS*

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: Before analysis**; **Shell Parameters**: toggle on **Specify transverse shear**

Use the following option for a homogeneous or composite shell section:

Property module: shell section editor: **Section integration: Before analysis**; **Advanced**: toggle on **Specify transverse shear**

Defining the Initial Section Forces and Moments

You can define initial stresses (see *Defining Initial Stresses*) for general shell sections that will be applied as initial section forces and moments. Initial conditions can be specified only for the membrane forces, the bending moments, and the twisting moment. Initial conditions cannot be prescribed for the transverse shear forces.

Specifying the Order of Accuracy in the Abaqus/Explicit Shell Element Formulation

In Abaqus/Explicit you can specify second-order accuracy in the shell element formulation. See *Section Controls* for more information.

Input File Usage: **SHELL GENERAL SECTION*, CONTROLS=*name*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Element Controls**

Specifying Nondefault Hourglass Control Parameters for Reduced-Integration Shell Elements

You can specify a nondefault hourglass control formulation or scale factors for elements that use reduced integration. See *Section Controls* for more information.

In Abaqus/Standard the nondefault enhanced hourglass control formulation is available only for S4R and SC8R elements.

In Abaqus/Standard you can modify the default values for hourglass control stiffness based on the default total stiffness approach for elements that use hourglass control and define a scaling factor for the stiffness associated with the drill degree of freedom (rotation about the surface normal) for elements that use six degrees of freedom at a node.

No default values are available for hourglass control stiffness if the section properties are specified in user subroutine *UGENS*. Therefore, you must specify the hourglass control stiffness when *UGENS* is used to specify the section properties for reduced-integration elements.

The stiffness associated with the drill degree of freedom is the average of the direct components of the transverse shear stiffness multiplied by a scaling factor. In most cases the default scaling factor is appropriate for constraining the drill rotation to follow the in-plane rotation of the element. If an additional scaling factor is defined, the

additional scaling factor should not increase or decrease the drill stiffness by more than a factor of 100.0 for most typical applications. Usually, a scaling factor between 0.1 and 10.0 is appropriate.

There are no hourglass stiffness factors or scale factors for hourglass stiffness for the nondefault enhanced hourglass control formulation. You can define the scale factor for the drill stiffness for the nondefault enhanced hourglass control formulation.

Input File Usage: Use both of the following options to specify a nondefault hourglass control formulation or scale factors for reduced-integration elements:

**SECTION CONTROLS*, NAME=*name*
**SHELL GENERAL SECTION*, CONTROLS=*name*

Use both of the following options in Abaqus/Standard to modify the default values for hourglass control stiffness based on the default total stiffness approach for reduced-integration elements and to define a scaling factor for the stiffness associated with the drill degree of freedom (rotation about the surface normal) for six degree of freedom elements:

**SHELL GENERAL SECTION*
**HOURGLASS STIFFNESS*

Abaqus/CAE Usage: Mesh module: **Mesh->Element Type**: **Element Controls**

Defining Density for Conventional Shells

You can define the mass per unit area for conventional shell elements whose section properties are specified directly in terms of the section stiffness (either directly in the section definition or, in Abaqus/Standard, in user subroutine *UGENS*). The density is required, for example, in a dynamic analysis or for gravity loading. See *Density* for details.

The density is defined as part of the material definition for shells whose section properties include a material definition.

This functionality is similar to the more general functionality of defining a nonstructural mass contribution (see *Nonstructural Mass Definition*.) The only difference between the two definitions is that the nonstructural mass contributes to the rotary inertia terms about the midsurface while the additional mass defined in the section definition does not.

Input File Usage: Use the following option to define the density directly:

**SHELL GENERAL SECTION*, ELSET=*name*, DENSITY=ρ

Use the following option in Abaqus/Standard to define the density in user subroutine *UGENS*:

**SHELL GENERAL SECTION*, ELSET=*name*, USER,
DENSITY=ρ

Abaqus/CAE Usage: Use the following option for a composite layup:

Property module: composite layup editor: **Section integration: Before analysis**; **Shell Parameters**: toggle on **Density**, and enter ρ

Use the following option for a homogeneous or composite shell section:

Property module: shell section editor: **Section integration: Before analysis**; **Advanced**: toggle on **Density**, and enter ρ

You cannot define the shell section properties in user subroutine *UGENS* in Abaqus/CAE.

Defining Damping

You can include mass and stiffness proportional damping in a shell section definition. See *Material Damping* for more information about material damping in Abaqus.

Specifying Temperature and Field Variables

Temperatures and field variables can be specified by defining the value at the reference surface of the shell or by defining the values at the nodes of a continuum shell element. The actual values of the temperatures and field variables are specified as either predefined fields or initial conditions (see *Predefined Fields* or *Initial Conditions*).

Output

The following output variables are available from Abaqus/Explicit as element output: section forces and moments, section strains, element energies, element stable time increment, and element mass scaling factor.

The output that is available from Abaqus/Standard depends on how the section behavior is defined.

Output if the section is defined in terms of material properties

For shells whose section properties include a material definition (homogeneous or composite), section forces and moments and section strains are available as element output. The section moments are calculated relative to the reference surface. In addition, stress (in-plane and, for certain elements, transverse shear), strain, and orthotropic failure measures can be output. Since the behavior of the material is linear, three section points per layer (the bottom, middle, and top, respectively) are available for output. Stress invariants and principal stresses are not available as output but can be visualized in Abaqus/CAE.

Output if the equivalent section properties are specified directly or in *UGENS*

If the $[\mathbf{D}]$ matrix is used to specify the equivalent section properties directly or if user subroutine *UGENS* is used, section point stresses and strains and section strains are not available for output or visualization inAbaqus/CAE; only section forces and moments can be requested for outputor visualized inAbaqus/CAE.

Three-Dimensional Conventional Shell Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the three-dimensional shell elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *About Shell Elements*
- *Choosing a Shell Element*
- **NODAL THICKNESS*
- **SHELL GENERAL SECTION*
- **SHELL SECTION*

Element Types

Stress/Displacement Elements

STRI3$^{(S)}$	3-node triangular facet thin shell
S3	3-node triangular general-purpose shell, finite membrane strains (identical to element S3R)
S3R	3-node triangular general-purpose shell, finite membrane strains (identical to element S3)
S3RS$^{(E)}$	3-node triangular shell, small membrane strains
STRI65$^{(S)}$	6-node triangular thin shell, using five degrees of freedom per node
S4	4-node general-purpose shell, finite membrane strains
S4R	4-node general-purpose shell, reduced integration with hourglass control, finite membrane strains
S4RS$^{(E)}$	4-node, reduced integration, shell with hourglass control, small membrane strains
S4RSW$^{(E)}$	4-node, reduced integration, shell with hourglass control, small membrane strains, warping considered in small-strain formulation
S4R5$^{(S)}$	4-node thin shell, reduced integration with hourglass control, using five degrees of freedom per node
S8R$^{(S)}$	8-node doubly curved thick shell, reduced integration
S8R5$^{(S)}$	8-node doubly curved thin shell, reduced integration, using five degrees of freedom per node
S9R5$^{(S)}$	9-node doubly curved thin shell, reduced integration, using five degrees of freedom per node

Active Degrees of Freedom

1, 2, 3, 4, 5, 6 for STRI3, S3R, S3RS, S4, S4R, S4RS, S4RSW, S8R

1, 2, 3 and two in-surface rotations for STRI65, S4R5, S8R5, S9R5 at most nodes

1, 2, 3, 4, 5, 6 for STRI65, S4R5, S8R5, S9R5 at any node that

- has a boundary condition on a rotational degree of freedom;
- is involved in a multi-point constraint that uses rotational degrees of freedom;
- is attached to a beam or to a shell element that uses six degrees of freedom at all nodes (such as S4R, S8R, STRI3, etc.);
- is a point where different elements have different surface normals (user-specified normal definitions or normal definitions created by Abaqus because the surface is folded); or
- is loaded with moments.

Additional Solution Variables

Element type S8R5 has three displacement and two rotation variables at an internally generated midbody node.

Heat Transfer Elements

DS3$^{(S)}$	3-node triangular shell
DS4$^{(S)}$	4-node quadrilateral shell
DS6$^{(S)}$	6-node triangular shell
DS8$^{(S)}$	8-node quadrilateral shell

Active Degrees of Freedom

11, 12, etc. (temperatures through the thickness as described in *Choosing a Shell Element*)

Additional Solution Variables

None.

Coupled Temperature-Displacement Elements

S3T$^{(S)}$	3-node triangular general-purpose shell, finite membrane strains, bilinear temperature in the shell surface (identical to element S3RT)
S3RT$^{(S)}$	3-node thermally coupled triangular general-purpose shell, finite membrane strains (identical to element S3T)
S3RT$^{(E)}$	3-node thermally-coupled triangular general-purpose shell, finite membrane strains

S4T$^{(S)}$	4-node general-purpose shell, finite membrane strains, bilinear temperature in the shell surface
S4RT	4-node thermally coupled general-purpose shell, reduced integration, hourglass control, finite membrane strains
S8RT$^{(S)}$	8-node thick shell, biquadratic displacement, bilinear temperature in the shell surface

Active Degrees of Freedom

1, 2, 3, 4, 5, 6 at all nodes

11, 12, 13, etc. (temperatures through the thickness as described in *Choosing a Shell Element*) at all nodes for S3T, S3RT, S4T, and S4RT; and at the corner nodes only for S8RT

Additional Solution Variables

None.

Nodal Coordinates Required

X, Y, Z and, optionally for shells with displacement degrees of freedom in Abaqus/Standard, N_x, N_y, N_z, the direction cosines of the shell normal at the node.

Element Property Definition

Input File Usage: Use either of the following options for stress/displacement elements:

**SHELL SECTION*
**SHELL GENERAL SECTION*

Use the following option for heat transfer or coupled temperature-displacement elements:

**SHELL SECTION*

In addition, use the following option for variable thickness shells:

**NODAL THICKNESS*

Abaqus/CAE Usage: Property module: **Create Section**: select **Shell** as the section **Category** and **Homogeneous** or **Composite** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are available for all elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*.

Body forces, centrifugal loads, and Coriolis forces must be given as force per unit area if the equivalent section properties are specified directly as part of the general shell section definition.

Load ID (*DLOAD): BX
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force (give magnitude as force per unit volume) in the global *X*-direction.

Load ID (*DLOAD): BY
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force (give magnitude as force per unit volume) in the global *Y*-direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force (give magnitude as force per unit volume) in the global *Z*-direction.

Load ID (*DLOAD): BXNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force (give magnitude as force per unit volume) in the global *X*-direction, with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BYNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force (give magnitude as force per unit volume) in the global *Y*-direction, with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force (give magnitude as force per unit volume) in the global *Z*-direction, with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): CENT (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-4} ($ML^{-3}T^{-2}$)

Description: Centrifugal load (magnitude defined as $\rho\omega^2$, where ρ is the mass density and ω is the angular speed).

Load ID (*DLOAD): CENTRIF (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular speed).

Load ID (*DLOAD): CORIO (S)
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-4}T$ ($ML^{-3}T^{-1}$)

Description: Coriolis force (magnitude input $\rho\omega$, where ρ is the mass density and ω is the angular speed). The load stiffness due to Coriolis loading is not accounted for in direct steady-state dynamics analysis.

Load ID (*DLOAD): EDLD*n*
Abaqus/CAE Load/Interaction: Shell edge load
Units: FL^{-1}
Description: General traction on edge *n*.

Load ID (*DLOAD): EDLD*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Nonuniform general traction on edge *n* with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): EDMOM*n*
Abaqus/CAE Load/Interaction: Shell edge load
Units: F
Description: Moment on edge *n*.

Load ID (*DLOAD): EDMOM*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Nonuniform moment on edge *n* with magnitude supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): EDNOR*n*
Abaqus/CAE Load/Interaction: Shell edge load
Units: FL^{-1}
Description: Normal traction on edge *n*.

Load ID (*DLOAD): EDNOR*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Nonuniform normal traction on edge *n* with magnitude supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): EDSHR*n*
Abaqus/CAE Load/Interaction: Shell edge load
Units: FL^{-1}
Description: Shear traction on edge *n*.

Load ID (*DLOAD): EDSHR*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Nonuniform shear traction on edge *n* with magnitude supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): EDTRA*n*
Abaqus/CAE Load/Interaction: Shell edge load
Units: FL^{-1}
Description: Transverse traction on edge *n*.

Load ID (*DLOAD): EDTRA*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Nonuniform transverse traction on edge *n* with magnitude supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): HP (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Hydrostatic pressure applied to the element reference surface and linear in global *Z*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure applied to the element reference surface. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): PNU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure applied to the element reference surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): ROTA (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}
Description: Rotary acceleration load (magnitude is input as $\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is the rotary acceleration).

Load ID (*DLOAD): ROTDYNF (S)
Abaqus/CAE Load/Interaction: Not supported
Units: T^{-1}
Description: Rotordynamic load (magnitude is input as $\boldsymbol{\omega}$, where $\boldsymbol{\omega}$ is the angular velocity).

Load ID (*DLOAD): SBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-5}T$
Description: Stagnation body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): SP (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T^2$
Description: Stagnation pressure applied to the element reference surface.

Load ID (*DLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element reference surface.

Load ID (*DLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): VBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T$
Description: Viscous body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): VP (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-3}T$
Description: Viscous surface pressure. The viscous pressure is proportional to the velocity normal to the element face and opposing the motion.

Foundations

Foundations are available for Abaqus/Standard elements with displacement degrees of freedom. They are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): F (S)
Abaqus/CAE Load/Interaction: Elastic foundation
Units: FL^{-3}
Description: Elastic foundation in the direction of the shell normal.

Distributed Heat Fluxes

Distributed heat fluxes are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DFLUX): BF (S)
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}\,T^{-1}$
Description: Body heat flux per unit volume.

Load ID (*DFLUX): BFNU (S)
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}\,T^{-1}$
Description: Nonuniform body heat flux per unit volume with magnitude supplied via user subroutine *DFLUX*.

Load ID (*DFLUX): MBFNU (S)
Abaqus/CAE Load/Interaction: Not supported

Units: JT^{-1}

Description: Nonuniform moving or stationary concentrated heat fluxes with magnitudes supplied via user subroutine *UMDFLUX*.

Load ID (*DFLUX): SNEG (S)

Abaqus/CAE Load/Interaction: Surface heat flux

Units: $JL^{-2}T^{-1}$

Description: Surface heat flux per unit area into the bottom face of the element.

Load ID (*DFLUX): SPOS (S)

Abaqus/CAE Load/Interaction: Surface heat flux

Units: $JL^{-2}T^{-1}$

Description: Surface heat flux per unit area into the top face of the element.

Load ID (*DFLUX): SNEGNU (S)

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}T^{-1}$

Description: Nonuniform surface heat flux per unit area into the bottom face of the element with magnitude supplied via user subroutine *DFLUX*.

Load ID (*DFLUX): SPOSNU (S)

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}T^{-1}$

Description: Nonuniform surface heat flux per unit area into the top face of the element with magnitude supplied via user subroutine *DFLUX*.

Film Conditions

Film conditions are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*FILM): FNEG (S)

Abaqus/CAE Load/Interaction: Surface film condition

Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Film coefficient and sink temperature (units of θ) provided on the bottom face of the element.

Load ID (*FILM): FPOS (S)

Abaqus/CAE Load/Interaction: Surface film condition

Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Film coefficient and sink temperature (units of θ) provided on the top face of the element.

Load ID (*FILM): FNEGNU (S)

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of θ) provided on the bottom face of the element with magnitude supplied via user subroutine *FILM*.

Load ID (*FILM): FPOSNU (S)

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of θ) provided on the top face of the element with magnitude supplied via user subroutine *FILM*.

Load ID (*FILM): FFS (S)
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Film coefficient and sink temperature (units of θ) provided on the top and bottom faces of the element.

Load ID (*FILM): FFSNU (S)
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of θ) provided on the top and bottom faces of the element with magnitude supplied via user subroutine.

Radiation Types

Radiation conditions are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*RADIATE): RNEG (S)
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) provided for the bottom face of the shell.

Load ID (*RADIATE): RPOS (S)
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) provided for the top face of the shell.

Load ID (*RADIATE): RFS (S)
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) provided for the top and bottom faces of the shell.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are available for all elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): EDLD
Abaqus/CAE Load/Interaction: Shell edge load
Units: FL^{-1}

Description: General traction on edge-based surface.

Load ID (*DSLOAD): EDLDNU (S)
Abaqus/CAE Load/Interaction: Shell edge load
Units: FL^{-1}

Description: Nonuniform general traction on edge-based surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): EDMOM
Abaqus/CAE Load/Interaction: Shell edge load
Units: F
Description: Moment on edge-based surface.

Load ID (*DSLOAD): EDMOMNU (S)
Abaqus/CAE Load/Interaction: Shell edge load
Units: F
Description: Nonuniform moment on edge-based surface with magnitude supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): EDNOR
Abaqus/CAE Load/Interaction: Shell edge load
Units: FL^{-1}
Description: Normal traction on edge-based surface.

Load ID (*DSLOAD): EDNORNU (S)
Abaqus/CAE Load/Interaction: Shell edge load
Units: FL^{-1}
Description: Nonuniform normal traction on edge-based surface with magnitude supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): EDSHR
Abaqus/CAE Load/Interaction: Shell edge load
Units: FL^{-1}
Description: Shear traction on edge-based surface.

Load ID (*DSLOAD): EDSHRNU (S)
Abaqus/CAE Load/Interaction: Shell edge load
Units: FL^{-1}
Description: Nonuniform shear traction on edge-based surface with magnitude supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): EDTRA
Abaqus/CAE Load/Interaction: Shell edge load
Units: FL^{-1}
Description: Transverse traction on edge-based surface.

Load ID (*DSLOAD): EDTRANU (S)
Abaqus/CAE Load/Interaction: Shell edge load
Units: FL^{-1}
Description: Nonuniform transverse traction on edge-based surface with magnitude supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): HP (S)
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Hydrostatic pressure on the element reference surface and linear in global *Z*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): P

Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}

Description: Pressure on the element reference surface. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}

Description: Nonuniform pressure on the element reference surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): SP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-4}T^2$

Description: Stagnation pressure applied to the element reference surface.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Shear traction on the element reference surface.

Load ID (*DSLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: General traction on the element reference surface.

Load ID (*DSLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): VP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-3}T$

Description: Viscous surface pressure. The viscous pressure is proportional to the velocity normal to the element face and opposing the motion.

Distributed Heat Fluxes

Surface-based distributed heat fluxes are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DSFLUX): S (S)
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}\,T^{-1}$

Description: Surface heat flux per unit area into the element surface.

Load ID (*DSFLUX): SNU (S)
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Nonuniform surface heat flux per unit area into the element surface with magnitude supplied via user subroutine *DFLUX*.

Film Conditions

Surface-based film conditions are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*SFILM): F (S)
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\theta^{-1}$
Description: Film coefficient and sink temperature (units of θ) provided on the element surface.

Load ID (*SFILM): FNU (S)
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\theta^{-1}$
Description: Nonuniform film coefficient and sink temperature (units of θ) provided on the element surface with magnitude supplied via user subroutine *FILM*.

Radiation Types

Surface-based radiation conditions are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*SRADIATE): R (S)
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless
Description: Emissivity and sink temperature (units of θ) provided for the element surface.

Incident Wave Loading

Surface-based incident wave loads are available. They are specified as described in *Coupled Acoustic-Structural Analysis*. If the incident wave field includes a reflection off a plane outside the boundaries of the mesh, this effect can be included.

Element Output

If a local coordinate system is not assigned to the element, the stress/strain components, as well as the section forces/strains, are in the default directions on the surface defined by the convention given in *Conventions*. If a local coordinate system is assigned to the element through the section definition (*Orientations*), the stress/strain components and the section forces/strains are in the surface directions defined by the local coordinate system.

In large-displacement problems with elements that allow finite membrane strains in Abaqus/Standard and in all problems in Abaqus/Explicit, the local directions defined in the reference configuration are rotated into the current configuration by the average material rotation.

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11
Local **11** direct stress.

S22
Local **22** direct stress.

S12
Local **12** shear stress.

Section Forces, Moments, and Transverse Shear Forces

Available for elements with displacement degrees of freedom.

SF1
Direct membrane force per unit width in local 1-direction.

SF2
Direct membrane force per unit width in local 2-direction.

SF3
Shear membrane force per unit width in local 1–2 plane.

SF4
Transverse shear force per unit width in local 1-direction (available only for S3/S3R, S3RS, S4, S4R, S4RS, S4RSW, S8R, and S8RT).

SF5
Transverse shear force per unit width in local 2-direction (available only for S3/S3R, S3RS, S4, S4R, S4RS, S4RSW, S8R, and S8RT).

SM1
Bending moment force per unit width about local 2-axis.

SM2
Bending moment force per unit width about local 1-axis.

SM3
Twisting moment force per unit width in local 1–2 plane.

The section force and moment resultants per unit length in the normal basis directions in a given shell section of thickness h can be defined on this basis as

$$(SF1, SF2, SF3, SF4, SF5) = \int_{-h/2-z_0}^{h/2-z_0} (\sigma_{11}, \sigma_{22}, \sigma_{12}, \sigma_{13}, \sigma_{23})\, dz,$$

$$(SM1, SM2, SM3) = \int_{-h/2-z_0}^{h/2-z_0} (\sigma_{11}, \sigma_{22}, \sigma_{12})\, z\, dz,$$

where z_0 is the offset of the reference surface from the midsurface.

The section force SF6, which is the integral of σ_{33} through the shell thickness, is reported only for finite-strain shell elements and is zero because of the plane stress constitutive assumption. The total number of attributes written to the results file for finite-strain shell elements is 9; SF6 is the sixth attribute.

Average Section Stresses

Available for elements with displacement degrees of freedom.

SSAVG1 Average membrane stress in local 1-direction.

SSAVG2 Average membrane stress in local 2-direction.

SSAVG3 Average membrane stress in local 1–2 plane.

SSAVG4 Average transverse shear stress in local 1-direction.

SSAVG5 Average transverse shear stress in local 2-direction.

The average section stresses are defined as

$$(SSAVG1, SSAVG2, SSAVG3, SSAVG4, SSAVG5) = (SF1, SF2, SF3, SF4, SF5)/h,$$

where h is the current section thickness.

Section Strains, Curvatures, and Transverse Shear Strains

Available for elements with displacement degrees of freedom.

SE1 Direct membrane strain in local 1-direction.

SE2 Direct membrane strain in local 2-direction.

SE3 Shear membrane strain in local 1–2 plane.

SE4 Transverse shear strain in the local 1-direction (available only for S3/S3R, S3RS, S4, S4R, S4RS, S4RSW, S8R, and S8RT).

SE5 Transverse shear strain in the local 2-direction (available only for S3/S3R, S3RS, S4, S4R, S4RS, S4RSW, S8R, and S8RT).

SE6 Strain in the thickness direction (available only for S3/S3R, S3RS, S4, S4R, S4RS, and S4RSW).

SK1 Curvature change about local 2-axis.

SK2 Curvature change about local 1-axis.

SK3 Surface twist in local 1–2 plane.

The local directions are defined in *About Shell Elements*.

Shell Thickness

STH Shell thickness, which is the current section thickness for S3/S3R, S3RS, S4, S4R, S4RS, and S4RSW elements.

Transverse Shear Stress Estimates

Available for S3/S3R, S3RS, S4, S4R, S4RS, S4RSW, S8R, and S8RT elements.

TSHR13 13-component of transverse shear stress.

TSHR23 23-component of transverse shear stress.

Estimates of the transverse shear stresses are available at section integration points as output variables TSHR13 or TSHR23 for both Simpson's rule and Gauss quadrature. For Simpson's rule output of variables TSHR13 or TSHR23 should be requested at nondefault section points, since the default output is at section point 1 of the shell section where the transverse shear stresses vanish. For the small-strain elements in Abaqus/Explicit, transverse shear stress distributions are assumed constant for noncomposite sections and piecewise constant for composite sections; therefore, transverse shear stresses at integration points should be interpreted accordingly.

For element type S4 the transverse shear calculation is performed at the center of the element and assumed constant over the element. Hence, transverse shear strain, force, and stress will not vary over the area of the element.

For numerically integrated shell sections (with the exception of small-strain shells in Abaqus/Explicit), estimates of the interlaminar shear stresses in composite sections—i.e., the transverse shear stresses at the interface between two composite layers—can be obtained only by using Simpson's rule. With Gauss quadrature no section integration point exists at the interface between composite layers.

Unlike the S11, S22, and S12 in-plane stress components, transverse shear stress components TSHR13 and TSHR23 are not calculated from the constitutive behavior at points through the shell section. They are estimated by matching the elastic strain energy associated with shear deformation of the shell section with that based on piecewise quadratic variation of the transverse shear stress across the section, under conditions of bending about one axis (see *Transverse shear stiffness in composite shells and offsets from the midsurface*). Therefore, interlaminar shear stress calculation is supported only when each layer of the shell section is defined using an elastic material model or a user-defined material model with the elastic transverse shear moduli defined. If you specify the transverse shear stiffness values, interlaminar shear stress output is not available.

Heat Flux Components

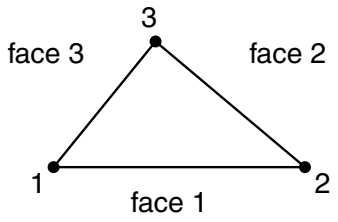
Available for elements with temperature degrees of freedom.

HFL1 Heat flux in local 1-direction.

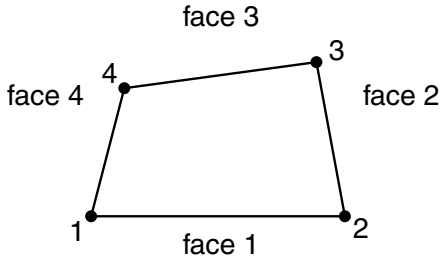
HFL2 Heat flux in local 2-direction.

HFL3 Heat flux in local 3-direction.

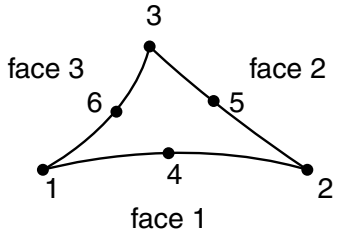
Node Ordering on Elements



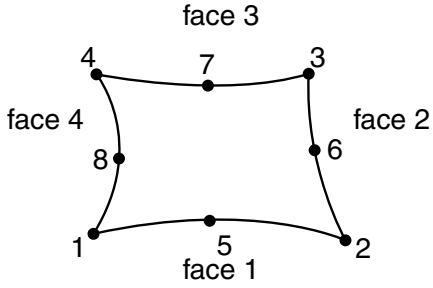
3-node element



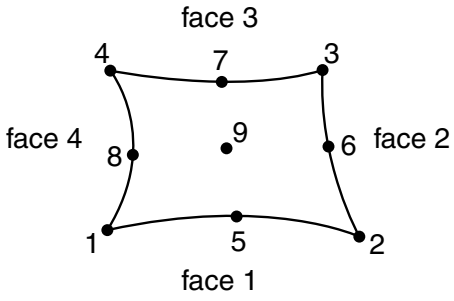
4-node element



6-node element



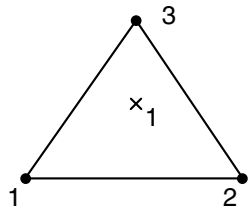
8-node element



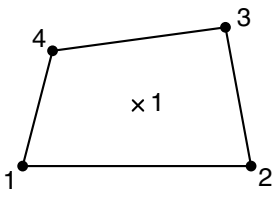
9-node element

Numbering of Integration Points for Output

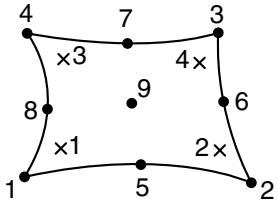
Stress/Displacement Analysis



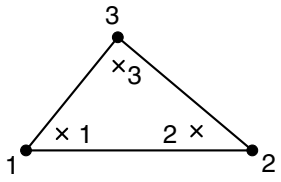
S3R element



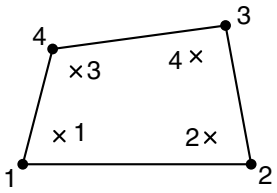
4-node reduced integration element



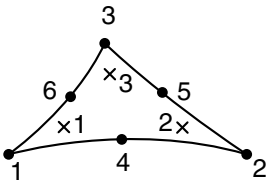
9-node reduced integration element



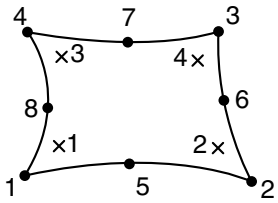
STRI3 element



4-node full integration element

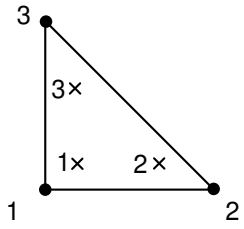


6-node element

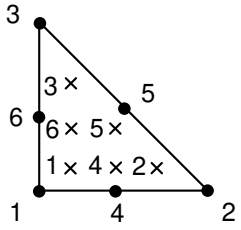


8-node reduced integration element

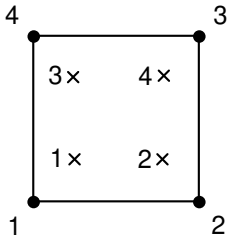
Heat Transfer Analysis



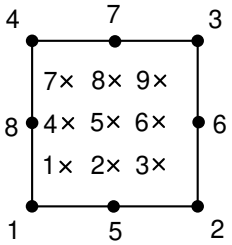
DS3



DS6



DS4



DS8

Continuum Shell Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the continuum shell elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *About Shell Elements*
- *Choosing a Shell Element*
- **SHELL GENERAL SECTION*
- **SHELL SECTION*

Element Types

Stress/Displacement Elements

SC6R	6-node triangular (in-plane) prism continuum shell, general-purpose, finite membrane strains
SC8R	8-node hexahedron, general-purpose, finite membrane strains

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

None.

Coupled Temperature-Displacement Elements

SC6RT	6-node linear displacement and temperature, triangular (in-plane) prism continuum shell, general-purpose continuum shell, finite membrane strains
SC8RT	8-node linear displacement and temperature, hexahedron, general-purpose continuum shell, finite membrane strains

Active Degrees of Freedom

1, 2, 3, 11

Additional Solution Variables

None.

Nodal Coordinates Required

X, Y, Z

Element Property Definition

Input File Usage: Use either of the following options:

**SHELL SECTION*
**SHELL GENERAL SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Shell** as the section **Category** and **Homogeneous** or **Composite** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force (give magnitude as force per unit volume) in the global *X*-direction.

Load ID (*DLOAD): BY
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force (give magnitude as force per unit volume) in the global *Y*-direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force (give magnitude as force per unit volume) in the global *Z*-direction.

Load ID (*DLOAD): BXNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force (give magnitude as force per unit volume) in the global *X*-direction, with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BYNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}

Description: Nonuniform body force (give magnitude as force per unit volume) in the global *Y*-direction, with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force (give magnitude as force per unit volume) in the global *Z*-direction, with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): CENT (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-4} ($ML^{-3}T^{-2}$)

Description: Centrifugal load (magnitude defined as $\rho\omega^2$, where ρ is the mass density and ω is the angular speed).

Load ID (*DLOAD): CENTRIF (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular speed).

Load ID (*DLOAD): CORIO (S)
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-4}T$ ($ML^{-3}T^{-1}$)

Description: Coriolis force (magnitude input $\rho\omega$, where ρ is the mass density and ω is the angular speed). The load stiffness due to Coriolis loading is not accounted for in direct steady-state dynamics analysis.

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): HP*n* (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Hydrostatic pressure on face *n*, linear in global *Z*. A positive pressure is directed into the element.

Load ID (*DLOAD): P*n*
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on face *n*. A positive pressure is directed into the element.

Load ID (*DLOAD): P*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure on face *n* with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. A positive pressure is directed into the element.

Load ID (*DLOAD): ROTA (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}
Description: Rotary acceleration load (magnitude is input as α, where α is the rotary acceleration).

Load ID (*DLOAD): ROTDYNF (S)
Abaqus/CAE Load/Interaction: Not supported
Units: T^{-1}
Description: Rotordynamic load (magnitude is input as ω, where ω is the angular velocity).

Load ID (*DLOAD): SBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-5}T^2$
Description: Stagnation body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): SP*n* (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T^2$
Description: Stagnation pressure on face *n*.

Load ID (*DLOAD): TRSHR*n*
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on face *n*.

Load ID (*DLOAD): TRSHR*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform shear traction on face *n* with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC*n*
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on face *n*.

Load ID (*DLOAD): TRVEC*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform general traction on face *n* with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): VBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T$
Description: Viscous body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): VP*n* (E)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^3T
Description: Viscous pressure on face *n*, applying a pressure proportional to the velocity normal to the face and opposing the motion.

Foundations

Foundations are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): F*n* (S)

Abaqus/CAE Load/Interaction: Elastic foundation
Units: FL^{-3}
Description: Elastic foundation on face *n*. A positive pressure is directed into the element.

Distributed Heat Fluxes

Distributed heat fluxes are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DFLUX): BF
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Heat body flux per unit volume.

Load ID (*DFLUX): BFNU (S)
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Nonuniform heat body flux per unit volume with magnitude supplied via user subroutine *DFLUX*.

Load ID (*DFLUX): S*n*
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Heat surface flux per unit area into face *n*.

Load ID (*DFLUX): S*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}$
Description: Nonuniform heat surface flux per unit area into face *n* with magnitude supplied via user subroutine *DFLUX*.

Film Conditions

Film conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*FILM): F*n*
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\theta^{-1}$
Description: Film coefficient and sink temperature (units of θ) provided on face *n*.

Load ID (*FILM): F*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\theta^{-1}$
Description: Nonuniform film coefficient and sink temperature (units of θ) provided on face *n* with magnitude supplied via user subroutine *FILM*.

Radiation Types

Radiation conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*RADIATE): R*n*
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) provided on face *n*.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): HP (S)
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}

Description: Hydrostatic pressure applied to the element surface, linear in global *Z*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}

Description: Pressure applied to the element surface. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}

Description: Nonuniform pressure applied to the element surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): SP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-4}T^2$

Description: Stagnation pressure applied to the element reference surface.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Shear traction on the element reference surface.

Load ID (*DSLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: General traction on the element reference surface.

Load ID (*DSLOAD): TRVECNU (S)

Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): VP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{3}T$
Description: Viscous surface pressure. The viscous pressure is proportional to the velocity normal to the element face and opposing the motion.

Distributed Heat Fluxes

Surface-based heat fluxes are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DSFLUX): S
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Heat surface flux per unit area into the element surface.

Load ID (*DSFLUX): SNU (S)
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Nonuniform heat surface flux per unit area into the element surface with magnitude supplied via user subroutine *DFLUX*.

Film Conditions

Surface-based film conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*SFILM): F
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\boldsymbol{\theta}^{-1}$
Description: Film coefficient and sink temperature (units of $\boldsymbol{\theta}$) provided on the element surface.

Load ID (*SFILM): FNU (S)
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\boldsymbol{\theta}^{-1}$
Description: Nonuniform film coefficient and sink temperature (units of $\boldsymbol{\theta}$) provided on the element surface with magnitude supplied via user subroutine *FILM*.

Radiation Types

Surface-based radiation conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*SRADIATE): R
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless
Description: Emissivity and sink temperature (units of $\boldsymbol{\theta}$) provided on the element surface.

Element Output

If a local coordinate system is not assigned to the element, the stress/strain components, as well as the section forces/strains, are in the default directions on the surface defined by the convention given in *Conventions*. If a local coordinate system is assigned to the element through the section definition (*Orientations*), the stress/strain components and the section forces/strains are in the surface directions defined by the local coordinate system.

The local directions defined in the reference configuration are rotated into the current configuration by the average material rotation.

In the case of composite shells the components of section forces, section strains, and transverse shear stress estimates for stacked continuum shells (CTSHR13 and CTSHR23) are reported in the local orientation defined for the entire section (or the default shell coordinate directions if no section orientation is used). Components of stress, strain, and transverse shear stress (TSHR13 and TSHR23) are given with respect to the individual layer orientations.

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available. All tensors have the same components. For example, the stress components are as follows:

S11 Local **11** direct stress.

S22 Local **22** direct stress.

S12 Local **12** shear stress.

The stress in the thickness direction, σ_{33}, is reported as zero to the output database as discussed in *Abaqus/Standard Output Variable Identifiers*. σ_{33} might be obtained through the average section stress variable SSAVG6. Output of in-plane stress components of continuum shell elements does not include Poisson effects due to changes in the thickness direction.

Heat Flux Components

Available for elements with temperature degrees of freedom.

HFL1 Heat flux in the *X*-direction.

HFL2 Heat flux in the *Y*-direction.

HFL3 Heat flux in the *Z*-direction.

Section Forces, Moments, and Transverse Shear Forces

SF1 Direct membrane force per unit width in local 1-direction.

SF2 Direct membrane force per unit width in local 2-direction.

SF3 Shear membrane force per unit width in local 1–2 plane.

SF4 Transverse shear force per unit width in local 1-direction.

SF5 Transverse shear force per unit width in local 2-direction.

SF6 Thickness stress integrated over the element thickness.

SM1 Bending moment force per unit width about local 2-axis.

SM2 Bending moment force per unit width about local 1-axis.

SM3 Twisting moment force per unit width in local 1–2 plane.

The section force and moment resultants per unit length in the normal basis directions in a given shell section of thickness h can be defined on this basis as

$$(SF1, SF2, SF3, SF4, SF5, SF6) = \int_{-h/2}^{h/2} (\sigma_{11}, \sigma_{22}, \sigma_{12}, \sigma_{13}, \sigma_{23}, \sigma_{33})\, dz,$$

$$(SM1, SM2, SM3) = \int_{-h/2}^{h/2} (\sigma_{11}, \sigma_{22}, \sigma_{12})\, z\, dz.$$

where stress in the thickness direction σ_{33} is constant through the thickness. Outputs of in-plane section forces of continuum shell elements do not include Poisson effects due to changes in the thickness direction.

Average Section Stresses

SSAVG1 Average membrane stress in local 1-direction.

SSAVG2 Average membrane stress in local 2-direction.

SSAVG3 Average membrane stress in local 1–2 plane.

SSAVG4 Average transverse shear stress in local 1-direction.

SSAVG5 Average transverse shear stress in local 2-direction.

SSAVG6 Average thickness stress in the local 3-direction.

The average section stresses are defined as

$$SSAVGn = SFn/h$$

where $n = 1, \ldots, 6$ and h is the current section thickness. σ_{33} is constant through the thickness.

Section Strains, Curvatures, and Transverse Shear Strains

SE1 Direct membrane strain in local 1-direction.

SE2 Direct membrane strain in local 2-direction.

SE3 Shear membrane strain in local 1–2 plane.

SE4 Transverse shear strain in the local 1-direction.

SE5 Transverse shear strain in the local 2-direction.

SE6 Total strain in the thickness direction.

SK1 Curvature change about local 2-axis.

SK2 Curvature change about local 1-axis.

SK3 Surface twist in local 1–2 plane.

The local directions are defined in *About Shell Elements*.

Shell Thickness

STH Section thickness, which is the current section thickness if geometric nonlinearity is considered; otherwise, it is the initial section thickness.

Transverse Shear Stress Estimates

TSHR13 13-component of transverse shear stress.

TSHR23 23-component of transverse shear stress.

Estimates of the transverse shear stresses are available at section integration points as output variables TSHR13 or TSHR23 for both Simpson's rule and Gauss quadrature. For Simpson's rule output of variables TSHR13 or TSHR23 should be requested at nondefault section points, since the default output is at section point 1 of the shell section where the transverse shear stresses vanish.

For numerically integrated sections, estimates of the interlaminar shear stresses in composite sections—that is, the transverse shear stresses at the interface between two composite layers—can be obtained only by using Simpson's rule. With Gauss quadrature no section integration point exists at the interface between composite layers.

Unlike the S11, S22, and S12 in-surface stress components, TSHR13 and TSHR23 are not calculated from the constitutive behavior at points through the shell section. They are estimated by matching the elastic strain energy associated with shear deformation of the shell section with that based on piecewise quadratic variation of the transverse shear stress across the section, under conditions of bending about one axis (see *Transverse shear stiffness in composite shells and offsets from the midsurface*). Therefore, interlaminar shear stress calculation is supported only when each layer of the shell section is defined using an elastic material model or a user-defined material model with the elastic transverse shear moduli defined. If you specify the transverse shear stiffness values, interlaminar shear stress output is not available. TSHR13 and TSHR23 are valid only for sections that have one element through the thickness direction. For sections with two or more continuum shell elements stacked in the thickness direction, output variables SSAVG4 and SSAVG5 or CTSHR13 and CTSHR23 should be used instead. An example using SSAVG4 and SSAVG5 to estimate the transverse shear stress distribution in stacked continuum shells can be found in *Composite shells in cylindrical bending*.

Transverse Shear Stress Estimates for Stacked Continuum Shells

CTSHR13	13-component of transverse shear stress for stacked continuum shells.
CTSHR23	23-component of transverse shear stress for stacked continuum shells.

Estimates of the transverse shear stresses that take into account the continuity of interlaminar transverse shear stress for stacked continuum shells are available at section integration points as output variables CTSHR13 or CTSHR23 for both Simpson's rule and Gauss quadrature. CTSHR13 or CTSHR23 are available only in Abaqus/Standard.

CTSHR13 and CTSHR23 are not calculated from the constitutive behavior at points through the shell section. They are estimated by assuming a quadratic variation of shear stress across the element section and by enforcing the continuity of interface transverse shear between adjoining continuum elements in a stack. It is also assumed that the transverse shear is zero at the free boundaries of a stack.

The intended use case for CTSHR13 and CTSHR23 is to estimate the through-the-thickness transverse shear stress for flat or nearly flat composite plates that are modeled with stacked continuum shell elements where each continuum element in the stack models a single material layer. Central to CTSHR13 and CTSHR23 is the concept of a “stack” of continuum shell elements.

During input file preprocessing Abaqus partitions all the continuum shells in a model into stacks. A “stack” is defined as a contiguous set of continuum shells whose first and last elements lie on a free boundary and who are connected through shared nodes on the top and bottom element surfaces (as determined by the elements' stack directions). In this context a “free boundary” is a top or bottom surface of a continuum shell element that is not connected through its nodes to another continuum shell element. For example, assuming that the stack direction of all the elements in *Figure 1* is in the z-direction, elements 1–6 would form a stack.

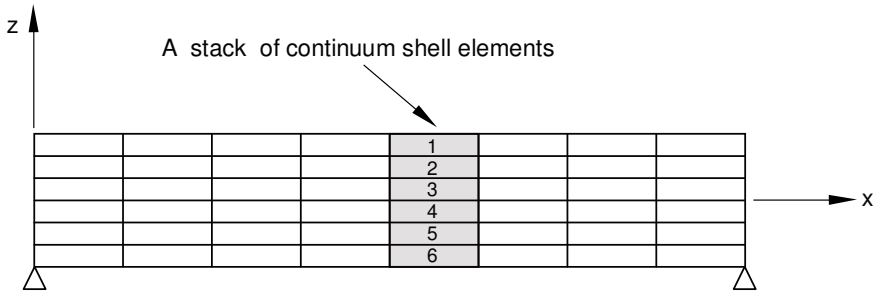


Figure 1: Composite plate meshed with six stacked continuum shells through the thickness.

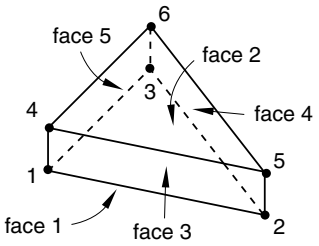
It is important to emphasize that stacks of continuum shells are connected through shared nodes, not through constraints or other elements. Suppose, for example, that in *Figure 1* element pairs 1–2, 2–3, 4–5, and 5–6 are connected to each other through shared nodes, but elements 3 and 4 are connected through a constraint (such as a tied constraint). In that case Abaqus would interpret the bottom surface of element 3 and the top surface of element 4 as free boundaries; therefore, elements 1–3 would form one stack, and elements 4–6 would form a second independent stack. For another example, suppose that element 4 is not a continuum shell element. In this case elements 1–3 would form one stack, and elements 5–6 would form another stack. In a final example, suppose the stack directions of elements 1–5 are in the global z-direction and the stack direction of element 6 is in the global x-direction. In this case elements 1–5 would form a stack separate from element 6. In the three cases just discussed the computed values of CTSHR13 and CTSHR23 would probably not be what you wanted. It is more likely that you want elements 1–6 to be in the same stack. It might be necessary to make changes in your model to achieve this. You can review the partitioning of the continuum shell elements into stacks in the data file by making a model definition data request.

The continuum shell elements in a stack must satisfy certain criteria; otherwise, Abaqus marks the stack as *invalid* with respect to computing CTSHR13 or CTSHR23. If a stack is marked as invalid, CTSHR13 or CTSHR23, if requested, are not computed and are set to zero for all continuum shell elements in that stack. If a continuum shell element does not have an elastic material model, if you specify the transverse shear for any element in the stack, or if the element is specified as rigid, that stack is marked as invalid. A stack is also marked as invalid if the normal of any element in a stack is not within 10° of the average normal for the stack. In addition, if a continuum shell element is removed during the analysis, the stack to which the element belongs is marked as invalid until the element is reactivated.

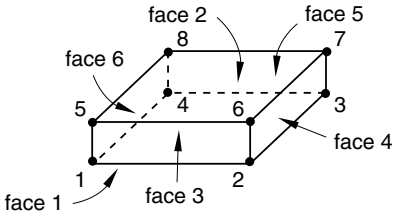
There are several other certain restrictions on CTSHR13 and CTSHR23. CTSHR13 and CTSHR23 are not available in any continuum shell element with a multi-layer composite material definition. However, having a multi-layer composite element in the stack does not invalidate the stack. For the purposes of computing CTSHR13 and CTSHR23, a maximum of 500 continuum shell elements can be put in any individual stack. If more than 500 continuum shell elements are stacked on top of each other, Abaqus issues a warning message during input file preprocessing, and CTSHR13 and CTSHR23 are not computed and are set to zero for all continuum shell elements in the model. CTSHR13 and CTSHR23 are not available if element operations are run in parallel (see *Parallel Execution in Abaqus/Standard*). CTSHR13 or CTSHR23 are currently available only for static and direct-integration dynamic analyses.

An example using CTSHR13 and CTSHR23 to estimate the transverse shear stress distribution in stacked continuum shells can be found in *Composite shells in cylindrical bending*.

Node Ordering on Elements



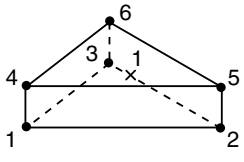
6-node continuum shell



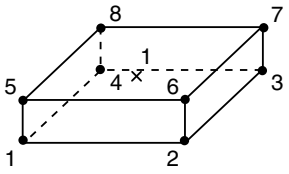
8-node continuum shell

Numbering of Integration Points for Output

Stress/Displacement Analysis



6-node continuum shell



8-node continuum shell

Axisymmetric Shell Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the axisymmetric shell elements available in Abaqus/Standard and Abaqus/Explicit. For axisymmetric shell geometries in which nonaxisymmetric behavior is expected, use the SAXA elements available in Abaqus/Standard (see *Axisymmetric Shell Elements with Nonlinear, Asymmetric Deformation*).

Conventions

Coordinate 1 is r, coordinate 2 is z. The r-direction corresponds to the global X-direction, and the z-direction corresponds to the global Y-direction. Coordinate 1 should be greater than or equal to zero.

Degree of freedom 1 is u_r, degree of freedom 2 is u_z, and degree of freedom 6 is rotation in the r–z plane.

Abaqus does not automatically apply any boundary conditions to nodes located along the symmetry axis. You should apply radial or symmetry boundary conditions on these nodes if desired.

Point loads and concentrated fluxes should be given as the value integrated around the circumference (that is, the load on the complete ring).

The meridional direction is the direction that is tangent to the element in the r–z plane; that is, the meridional direction is along the line that is rotated about the axis of symmetry to generate the full three-dimensional body.

The circumferential or hoop direction is the direction normal to the r–z plane.

References:

- *About Shell Elements*
- *Choosing a Shell Element*
- **NODAL THICKNESS*
- **SHELL GENERAL SECTION*
- **SHELL SECTION*

Element Types

Stress/Displacement Elements

SAX1	2-node linear axisymmetric thin or thick shell
SAX2$^{(S)}$	3-node thin or thick quadratic shell

Active Degrees of Freedom

1, 2, 6

Additional Solution Variables

None.

Heat Transfer Elements

DSAX1[(S)]	2-node shell
DSAX2[(S)]	3-node shell

Active Degrees of Freedom

11, 12, 13, etc. (temperatures through the thickness as described in *Choosing a Shell Element*)

Additional Solution Variables

None.

Coupled Temperature-Displacement Element

SAX2T[(S)]	3-node thin or thick shell, quadratic displacement, linear temperature in the shell surface

Active Degrees of Freedom

1, 2, 6 at all three nodes

11, 12, 13, etc. (temperatures through the thickness as described in *Choosing a Shell Element*) at the end nodes

Additional Solution Variables

None.

Nodal Coordinates Required

r, *z*, and optionally for shells with displacement degrees of freedom, N_r, N_z, the direction cosines of the shell normal at the node.

Element Property Definition

Input File Usage: Use either of the following options for stress/displacement elements:

**SHELL SECTION*
**SHELL GENERAL SECTION*

Use the following option for heat transfer or coupled temperature-displacement elements:

**SHELL SECTION*

In addition, use the following option for variable thickness shells:

**NODAL THICKNESS*

Abaqus/CAE Usage: Property module: **Create Section**: select **Shell** as the section **Category** and **Homogeneous** or **Composite** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are available for elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*.

Distributed load magnitudes are per unit area or per unit volume. They do not need to be multiplied by 2π.

Body forces and centrifugal loads must be given as force per unit area if a general shell section is used.

Load ID (*DLOAD): BR
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force per unit volume in the radial direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force per unit volume in the axial direction.

Load ID (*DLOAD): BRNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force per unit volume in the radial direction, with the magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force per unit volume in the global z-direction, with the magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): CENT$^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-4} ($ML^{-3}T^{-2}$)

Description: Centrifugal load (magnitude given as $\rho\omega^2$, where ρ is the mass density and ω is the angular velocity). Since only axisymmetric deformation is allowed, the spin axis must be the z-axis.

Load ID (*DLOAD): CENTRIF$^{(S)}$
Abaqus/CAE Load/Interaction: Rotational body force

Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity). Since only axisymmetric deformation is allowed, the spin axis must be the z-axis.

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude input as acceleration).

Load ID (*DLOAD): $HP^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Hydrostatic pressure applied to the element reference surface and linear in global *Z*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure applied to the element reference surface. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): PNU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure applied to the element reference surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): $SBF^{(E)}$
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-5}T^2$
Description: Stagnation body force in radial and axial directions.

Load ID (*DLOAD): $SP^{(E)}$
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T^2$
Description: Stagnation pressure applied to the element reference surface.

Load ID (*DLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DLOAD): $TRSHRNU^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element reference surface.

Load ID (*DLOAD): TRVECNU$^{(S)}$
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): VBF$^{(E)}$
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T$
Description: Viscous body force in radial and axial directions.

Load ID (*DLOAD): VP$^{(E)}$
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-3}T$
Description: Viscous surface pressure. The viscous pressure is proportional to the velocity normal to the element face and opposing the motion.

Foundations

Foundations are available for Abaqus/Standard elements with displacement degrees of freedom. They are specified as described in *Element Foundations*.

Load ID (*FOUNDATION): F$^{(S)}$
Abaqus/CAE Load/Interaction: Elastic foundation
Units: FL^{-3}
Description: Elastic foundation in the direction of the shell normal.

Distributed Heat Fluxes

Distributed heat fluxes are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DFLUX): BF$^{(S)}$
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}\,T^{-1}$
Description: Body heat flux per unit volume.

Load ID (*DFLUX): BFNU$^{(S)}$
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}\,T^{-1}$
Description: Nonuniform body heat flux per unit volume with magnitude supplied via user subroutine *DFLUX*.

Load ID (*DFLUX): SNEG$^{(S)}$
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}\,T^{-1}$
Description: Surface heat flux per unit area into the bottom face of the element.

Load ID (*DFLUX): SPOS$^{(S)}$
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}\,T^{-1}$
Description: Surface heat flux per unit area into the top face of the element.

Load ID (*DFLUX): SNEGNU$^{(S)}$

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}\,T^{-1}$

Description: Nonuniform surface heat flux per unit area into the bottom face of the element with magnitude supplied via user subroutine *DFLUX*.

Load ID (*DFLUX): SPOSNU$^{(S)}$

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}\,T^{-1}$

Description: Nonuniform surface heat flux per unit area into the top face of the element with magnitude supplied via user subroutine *DFLUX*.

Film Conditions

Film conditions are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*FILM): FNEG$^{(S)}$

Abaqus/CAE Load/Interaction: Surface film condition

Units: $JL^{-2}\,T^{-1}\boldsymbol{\theta}^{-1}$

Description: Film coefficient and sink temperature (units of $\boldsymbol{\theta}$) provided on the bottom face of the element.

Load ID (*FILM): FPOS$^{(S)}$

Abaqus/CAE Load/Interaction: Surface film condition

Units: $JL^{-2}\,T^{-1}\boldsymbol{\theta}^{-1}$

Description: Film coefficient and sink temperature (units of $\boldsymbol{\theta}$) provided on the top face of the element.

Load ID (*FILM): FNEGNU$^{(S)}$

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}\,T^{-1}\boldsymbol{\theta}^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of $\boldsymbol{\theta}$) provided on the bottom face of the element with magnitude supplied via user subroutine *FILM*.

Load ID (*FILM): FPOSNU$^{(S)}$

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}\,T^{-1}\boldsymbol{\theta}^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of $\boldsymbol{\theta}$) provided on the top face of the element with magnitude supplied via user subroutine *FILM*.

Radiation Types

Radiation conditions are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*RADIATE): RNEG$^{(S)}$

Abaqus/CAE Load/Interaction: Surface radiation

Units: Dimensionless

Description: Emissivity and sink temperature (units of $\boldsymbol{\theta}$) provided for the bottom face of the shell.

Load ID (*RADIATE): RPOS$^{(S)}$

Abaqus/CAE Load/Interaction: Surface radiation

Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) provided for the top face of the shell.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are available for elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*.

Distributed load magnitudes are per unit area or per unit volume. They do not need to be multiplied by 2π.

Load ID (*DSLOAD): HP$^{(S)}$
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Hydrostatic pressure on the element reference surface and linear in global *Z*. The pressure is positive in the direction opposite the surface normal.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element reference surface. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Nonuniform pressure on the element reference surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): SP$^{(E)}$
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-4}T^2$
Description: Stagnation pressure applied to the element reference surface.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DSLOAD): TRSHRNU$^{(S)}$
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element reference surface.

Load ID (*DSLOAD): TRVECNU$^{(S)}$

Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): $VP^{(E)}$
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-3}T$

Description: Viscous surface pressure. The viscous pressure is proportional to the velocity normal to the element surface and opposing the motion.

Distributed Heat Fluxes

Surface-based heat fluxes are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DSFLUX): $S^{(S)}$
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}\,T^{-1}$

Description: Surface heat flux per unit area into the element surface.

Load ID (*DSFLUX): $SNU^{(S)}$
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}\,T^{-1}$

Description: Nonuniform surface heat flux per unit area into the element surface with magnitude supplied via user subroutine *DFLUX*.

Film Conditions

Surface-based film conditions are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*SFILM): $F^{(S)}$
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}\,T^{-1}\boldsymbol{\theta}^{-1}$

Description: Film coefficient and sink temperature (units of $\boldsymbol{\theta}$) provided on the element surface.

Load ID (*SFILM): $FNU^{(S)}$
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}\,T^{-1}\boldsymbol{\theta}^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of $\boldsymbol{\theta}$) provided on the element surface with magnitude supplied via user subroutine *FILM*.

Radiation Types

Surface-based radiation conditions are available for elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*SRADIATE): $R^{(S)}$
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless

Description: Emissivity and sink temperature (units of $\boldsymbol{\theta}$) provided for the element surface.

Incident Wave Loading

Surface-based incident wave loads are available. They are specified as described in *Coupled Acoustic-Structural Analysis*. If the incident wave field includes a reflection off a plane outside the boundaries of the mesh, this effect can be included.

Element Output

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11 Meridional stress.

S22 Hoop (circumferential) stress.

Section Forces, Moments, and Transverse Shear Forces

Available for elements with displacement degrees of freedom.

SF1 Membrane force per unit width in the meridional direction.

SF2 Membrane force per unit width in the hoop direction.

SF3 Transverse shear force per unit width in the meridional direction (available only from Abaqus/Standard).

SF4 Integrated stress in the thickness direction; always zero (available only from Abaqus/Standard).

SM1 Bending moment per unit width about the hoop direction.

SM2 Bending moment per unit width about the meridional direction.

Section Strains, Curvature Changes, and Transverse Shear Strains

Available for elements with displacement degrees of freedom.

SE1 Membrane strain in the meridional direction.

SE2 Membrane strain in the hoop direction.

SE3 Transverse shear strain in the meridional direction (available only from Abaqus/Standard).

SE4 Strain in the thickness direction (available only from Abaqus/Standard).

SK1 Curvature change about the hoop direction.

SK2 Curvature change about the meridional direction.

Shell Thickness

STH Shell thickness, which is the current thickness for SAX1, SAX2, and SAX2T elements.

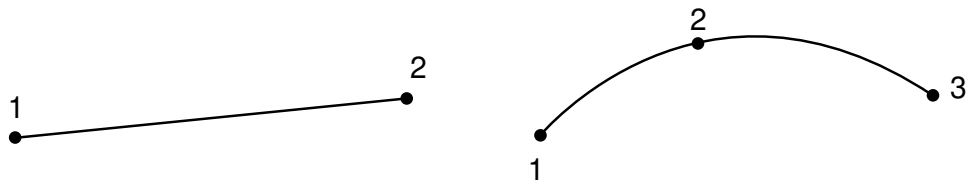
Heat Flux Components

Available for elements with temperature degrees of freedom.

HFL1 Heat flux in the meridional direction.

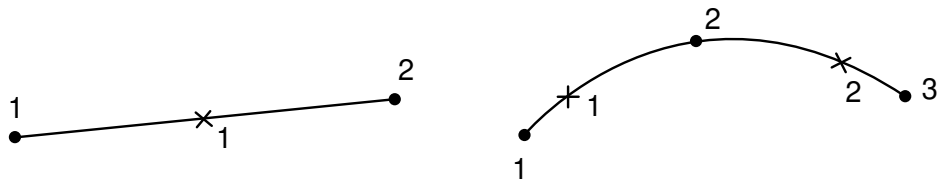
HFL2 Heat flux in the thickness direction.

Node Ordering on Elements



2 - node element 3 - node element

Numbering of Integration Points for Output



2 - node element 3 - node element

Axisymmetric Shell Elements with Nonlinear, Asymmetric Deformation

Products: Abaqus/Standard Abaqus/CAE

Overview

This section provides a reference to the axisymmetric shell elements with nonlinear, asymmetric deformation available in Abaqus/Standard. For an axisymmetric reference geometry where axisymmetric deformation is expected, use regular axisymmetric elements (see *Axisymmetric Shell Element Library*). For an axisymmetric reference geometry where nonaxisymmetric deformation is expected and the thickness to characteristic radius is high or through the thickness detail is required, use CAXA-type elements (see *Axisymmetric Solid Elements with Nonlinear, Asymmetric Deformation*).

Conventions

Coordinate 1 is r, coordinate 2 is z. The r-direction corresponds to the global X-direction in the $\theta = 0°$ plane and the global Y-direction in the $\theta = 90°$ plane, and the z-direction corresponds to the global Z-direction. Coordinate 1 should be greater than or equal to zero.

Degree of freedom 1 is u_r, degree of freedom 2 is u_z, degree of freedom 6 is rotation in the r–z plane.

Even though the symmetry in the r–z plane at $\theta = 0, \pi$ allows the modeling of half of the initially axisymmetric structure, the loading must be specified as the total load on the full axisymmetric body. Consider, for example, a cylindrical shell loaded by a unit uniform axial force. To produce a unit load on a SAXA element with four modes, the nodal forces

are 1/8, 1/4, 1/4, 1/4, and 1/8 at $\theta = 0, \pi/4, \pi/2, 3\pi/4$, and π, respectively.

The meridional direction is the direction tangent to the element in the r–z plane; that is, the meridional direction is along the line that is rotated about the axis of symmetry to generate the full three-dimensional body.

The circumferential or hoop direction is the direction normal to the r–z plane.

References:

- *About Shell Elements*
- *Choosing a Shell Element*
- **NODAL THICKNESS*
- **SHELL GENERAL SECTION*
- **SHELL SECTION*

Element Types

SAXA1*N*	Linear interpolation, Fourier shell element with 2 nodes in the meridional direction and *N* Fourier modes
SAXA2*N*	Quadratic interpolation, Fourier shell element with 3 nodes in the meridional direction and *N* Fourier modes

Active Degrees of Freedom

1, 2, 6

See *Figure 1* for the positive nodal displacement and rotation directions. The nodal rotation, ϕ_θ, is consistent with the SAX elements; however, a positive nodal rotation is in the negative θ-direction.

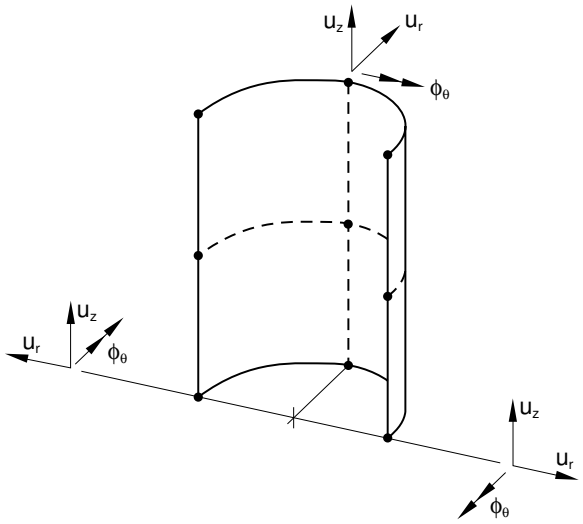


Figure 1: Element coordinate system and positive displacement/rotation directions. SAXA22 element shown.

Additional Solution Variables

SAXA$\mathbf{1}N$ elements have $\mathbf{6}N$ variables relating to (u_θ, ϕ_r, ϕ_z).

SAXA$\mathbf{2}N$ elements have $\mathbf{9}N$ variables relating to (u_θ, ϕ_r, ϕ_z).

Nodal Coordinates Required

r, z (given in the r–z plane for $\theta = 0$)

The two direction cosines, N_r and N_z, of the nodal normal field can be specified either in the nodal data or by a user-specified normal definition (see *Normal Definitions at Nodes*).

Element Property Definition

If a general shell section is used and the section stiffness matrix is given directly, a full 6 × 6 section stiffness should be specified (i.e., 21 constants as for a three-dimensional shell).

Input File Usage: Use either of the following options:

**SHELL SECTION*
**SHELL GENERAL SECTION*

In addition, use the following option for variable thickness shells:

**NODAL THICKNESS*

Abaqus/CAE Usage: Property module: **Create Section**: select **Shell** as the section **Category** and **Homogeneous** or **Composite** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Distributed load magnitudes are per unit area or per unit volume. They do not need to be multiplied by 2π times the radius.

Load ID (*DLOAD): BX
Units: FL^{-3}
Description: Body force per unit volume in the global *X*-direction.

Load ID (*DLOAD): BZ
Units: FL^{-3}
Description: Body force per unit volume in the global *Z*-direction.

Load ID (*DLOAD): BXNU
Units: FL^{-3}
Description: Nonuniform body force in the global *X*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BZNU
Units: FL^{-3}
Description: Nonuniform body force in the global *Z*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): HP
Units: FL^{-2}
Description: Hydrostatic pressure on the shell surface, linear in the global *Z*-direction.

Load ID (*DLOAD): P
Units: FL^{-2}
Description: Pressure on the shell surface.

Load ID (*DLOAD): PNU
Units: FL^{-2}
Description: Nonuniform pressure on the shell surface with magnitude supplied via user subroutine *DLOAD*.

Element Output

The numerical integration with respect to θ employs the trapezoidal rule. There are $2(N+1)$ equally spaced integration planes in the element, including the $\theta = 0°$ and $\theta = 180°$ planes, with *N* being the number of Fourier modes. Consequently, the radial nodal forces corresponding to pressure loads applied in the circumferential direction are distributed in this direction in the ratio of $1:1$ in the 1 Fourier mode element, $1:2:1$ in the 2 Fourier mode element, and $1:2:2:2:1$ in the 4 Fourier mode element. The sum of these consistent nodal forces is equal to the integral of the applied pressure over the full circumference (2π).

Stress, Strain, and Other Tensor Components

Stress and other tensors (including strain tensors) are available for elements with displacement degrees of freedom. All tensors have the same components. For example, the stress components are as follows:

S11 Meridional stress.

S22 Hoop (circumferential) stress.

S12 Local 12 shear stress (zero at $\theta = 0°$ and $180°$).

Section Forces

SF1 Direct membrane force per unit width in local 1-direction.

SF2 Direct membrane force per unit width in local 2-direction.

SF3 Shear membrane force per unit width in local 1–2 plane.

SF4 Integrated stress in the thickness direction; always zero.

SM1 Bending moment per unit width about local 2-axis.

SM2 Bending moment per unit width about local 1-axis.

SM3 Twisting moment per unit width in local 1–2 plane.

Section Strains

SE1 Direct membrane strain in local 1-direction.

SE2 Direct membrane strain in local 2-direction.

SE3 Shear membrane strain in local 1–2 plane.

SE4 Strain in the thickness direction.

SK1 Bending strain in local 1-direction.

SK2 Bending strain in local 2-direction.

SK3 Twisting strain in local 1–2 plane.

The section force and moment resultants per unit length in the normal basis directions for a given layer of thickness h can be defined, in components relative to this basis, as:

$$(\mathbf{SF1}, \mathbf{SF2}, \mathbf{SF3}) = \int_{-h/2-z_0}^{h/2-z_0} (\sigma_{11}, \sigma_{22}, \sigma_{12})\, dz,$$

$$(\mathbf{SM1}, \mathbf{SM2}, \mathbf{SM3}) = \int_{-h/2-z_0}^{h/2-z_0} (\sigma_{11}, \sigma_{22}, \sigma_{12})\, z\, dz,$$

where z_0 is the offset of the reference surface from the midsurface.

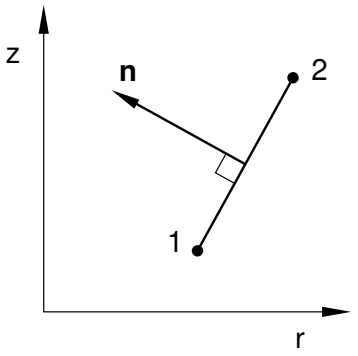
The local directions are defined in *Defining the Initial Geometry of Conventional Shell Elements*.

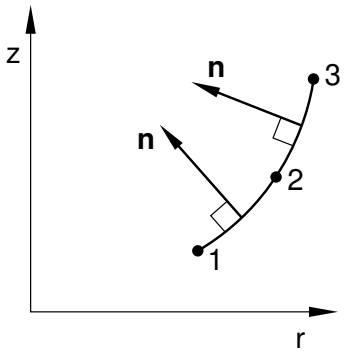
Current Shell Thickness

STH	Current shell thickness.

Node Ordering on Elements

The node ordering in the first generator plane ($\theta = 0$) of each element is shown below. You specify the line or curve of nodes in the generator plane just as with the SAX1 and SAX2 elements. Each element must have *N* more planes of nodes defined, where *N* is the number of Fourier modes used. Abaqus/Standard will generate these additional circumferential nodes and number them by adding a constant offset value to the nodes specified in the first plane (see *Element Definition*).





Shear Panel Elements

In this section:

- *Shear Panel Elements*
- *Shear Panel Element Library*

Shear Panel Elements

Products: Abaqus/Standard

References:

- *Shear Panel Element Library*
- **SHELL GENERAL SECTION*
- **DISTRIBUTION*

Overview

Shear panel elements:

- are 4-node surface elements that transmit only shear forces along the elements edges (no normal forces);
- have no bending stiffness;
- are typically used together with truss elements or beam elements that help stabilize the structure;
- can be used only with linear elastic material behavior;
- are intended to be used in small-displacement analysis only (large-rotation effects are not included);
- do not include thermal strain effects; and
- do not include material/element-based viscous damping effects;

Typical Applications

Shear panel elements are special surface elements for modeling thin reinforced plates and shells, such as thin skin panels in aircraft wings and fuselage structures. These elements can transmit shear forces only along their edges (see *Figure 1*). They cannot transmit normal forces, and they have no bending stiffness. The shear panel elements should not be used alone and should typically be used together with reinforcing truss or beam elements.

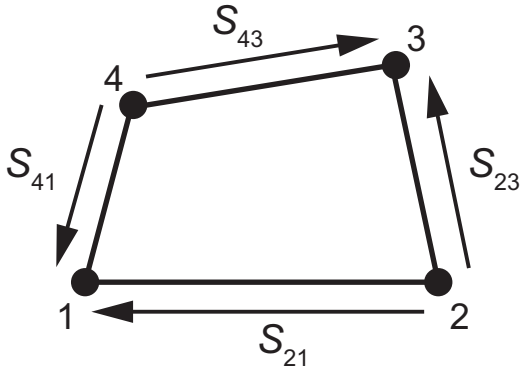


Figure 1: Shear flows along the edges of shear panel elements.

The elements have only displacement degrees of freedom. They are intended to be used only in small-displacement analyses. Large-rotation effects, thermal strain effects, and material/element-based viscous damping effects are not included. The elements can be used only with linear elastic material behavior.

Shear Panel Element Formulation

The shear panel element formulation is based on the work by Garvey (1951). For an arbitrary flat quadrilateral sheer panel element, the average shear flows along the element edges (see *Figure 1*) have the relationship

$$S_{21}S_{43} = S_{23}S_{41},$$

where S_{21}, S_{43}, S_{23}, and S_{41} denote the average shear flows along the element edges 12, 43, 23, and 41, respectively. The equivalent shear flow, q, of the element is defined as:

$$q^2 = S_{21}S_{43} = S_{23}S_{41}.$$

The shear strain, γ, associated with the equivalent shear flow, q, can be written as

$$\gamma = \frac{q}{Gt},$$

where G is the shear modulus, and t is the element thickness. The strain energy of the element can be expressed as

$$E = \frac{1}{2}\frac{q^2 A}{Gt},$$

where A is the element area. When the nodes of the element are not in a plane, the element calculations are based on nodes projected to a reference plane.

Element Normal Definition

The "top" surface of a shear panel element is the surface in the positive normal direction (see *Figure 2*) and is called the SPOS face for contact definition. The "bottom" surface is in the negative direction along the normal and is called the SNEG face for contact definition.

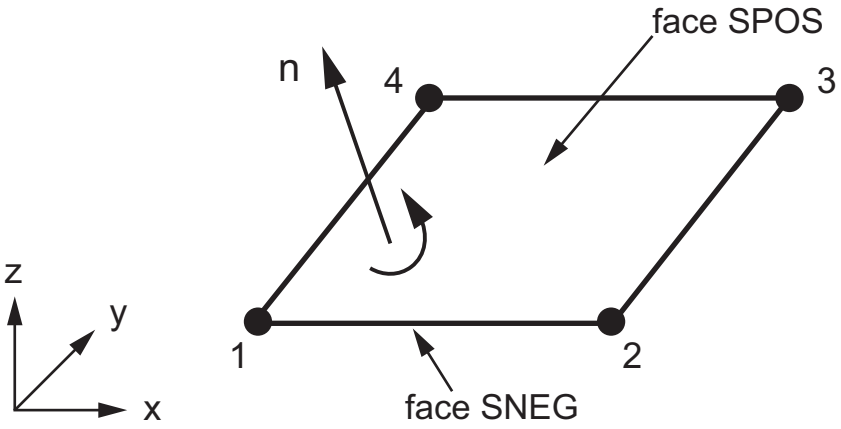


Figure 2: Positive normals for shear panel elements.

Defining the Element's Section Properties

The section properties of shear panel elements must be defined using a general shell section definition. You must associate these properties with a region of your model.

Input File Usage: **SHELL GENERAL SECTION*, ELSET=*name*

where the ELSET parameter refers to a set of shear panel elements.

Defining a Constant Section Thickness

You can define a constant section thickness as part of the section definition.

Input File Usage: **SHELL GENERAL SECTION*, ELSET=*name*
thickness

Defining a Variable Thickness Using Distributions

You can define a spatially varying thickness for shear panels using a distribution (*Distribution Definition*).

The distribution used to define shear panel thickness must have a default value. The default thickness is used by any shear panel element assigned to the shell general section that is not specifically assigned a value in the distribution.

Input File Usage: **SHELL GENERAL SECTION*, SHELL THICKNESS=*distribution name*

Assigning a Material Definition to a Set of Shear Panel Elements

You must associate a material definition with each general shell section definition.

Input File Usage: **SHELL GENERAL SECTION*, MATERIAL=*name*

Defining Density for Shear Panel Elements

You can define additional mass per unit area for shear panel elements directly in the section definition.

This functionality is similar to the more general functionality of defining a nonstructural mass contribution (see *Nonstructural Mass Definition*.) The only difference between the two definitions is that the nonstructural mass contributes to the rotary inertia terms about the midsurface while the additional mass defined in the section definition does not.

Input File Usage: Use the following option to define the density directly:

**SHELL GENERAL SECTION*, ELSET=*name*, DENSITY=ρ

References

- Garvey, S. J., “The Quadrilateral 'Shear' Panel: The Peculiar Stressing Problems Arising in the Structure of the Non-Rectangular Swept Wing,” *Aircraft Engineering*, vol. 23, pp. 134–144, 1951.

Shear Panel Element Library

Products: Abaqus/Standard

Overview

References:

- *Shear Panel Elements*
- **SHELL GENERAL SECTION*

Element Types

Stress/Displacement Elements

SHEAR4 4-node shear panel element

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage: **SHELL GENERAL SECTION*

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in the global *X*-direction.

Load ID (*DLOAD): BY

Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in the global *Y*-direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in the global *Z*-direction.

Load ID (*DLOAD): BXNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in the global *X*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BYNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in the global *Y*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in the global *Z*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): CENT
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-4} ($ML^{-3}T^{-2}$)

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): CENTRIF
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): CORIO
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-4}T$ ($ML^{-3}T^{-1}$)

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit volume, ω is the angular velocity). The load stiffness due to Coriolis loading is not accounted for in direct steady-state dynamic analysis.

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): HP
Abaqus/CAE Load/Interaction: Not supported

Units: FL^{-2}

Description: Hydrostatic pressure applied to the element reference surface and linear in global *Z*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}

Description: Pressure applied to the element reference surface. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): PNU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Nonuniform pressure applied to the element reference surface with magnitude supplied via user subroutine *DLOAD*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): ROTA
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Rotary acceleration load (magnitude is input as α, where α is the rotary acceleration).

Load ID (*DLOAD): ROTDYNF
Abaqus/CAE Load/Interaction: Not supported
Units: T^{-1}

Description: Rotordynamic load (magnitude is input as ω, where ω is the angular velocity).

Load ID (*DLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Shear traction on the element reference surface.

Load ID (*DLOAD): TRSHRNU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: General traction on the element reference surface.

Load ID (*DLOAD): TRVECNU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): HP
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Hydrostatic pressure on the element reference surface and linear in global *Z*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element reference surface. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Nonuniform pressure on the element reference surface with magnitude supplied via user subroutine *DLOAD*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DSLOAD): TRSHRNU
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element reference surface.

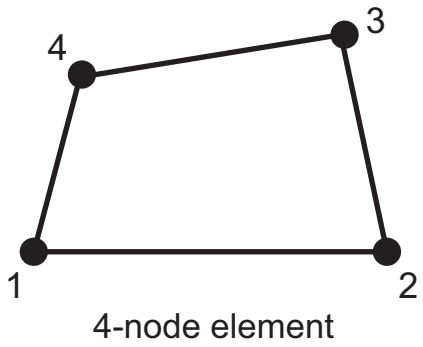
Load ID (*DSLOAD): TRVECNU
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Element Output

Equivalent shear flow is available for elements with displacement degrees of freedom.

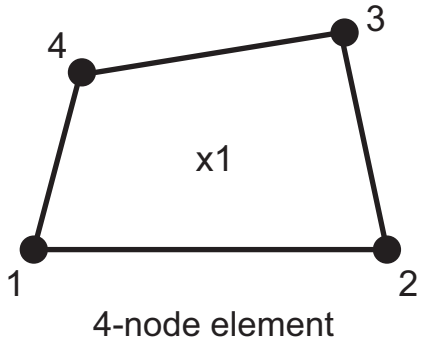
SQEQ	Equivalent shear flow.

Node Ordering on Elements



4-node element

Numbering of Integration Points for Output



4-node element

Inertial, Rigid, and Capacitance Elements

In this section:

- *Point Mass Elements*
- *Rotary Inertia Elements*
- *Rigid Elements*
- *Capacitance Elements*

Point Mass Elements

In this section:

- *Point Masses*
- *Mass Element Library*

Point Masses

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Mass Element Library*
- **MASS*
- *Defining point mass and rotary inertia*

Overview

Mass elements:

- allow the introduction of concentrated mass that is either isotropic or anisotropic at a point;
- are associated with the three translational degrees of freedom at a node.

If rotary inertia is also required (for example, to represent a rigid body), use element type ROTARYI (*Rotary Inertia*).

In addition to point masses, Abaqus provides a convenient nonstructural mass definition that can be used to smear mass from features that have negligible structural stiffness over a region that is typically adjacent to the nonstructural feature. The nonstructural mass can be specified in the form of a total mass value, a mass per unit volume, a mass per unit area, or a mass per unit length (see *Nonstructural Mass Definition*).

Defining the Isotropic Mass Value

You specify a mass magnitude, which is associated with the three translational degrees of freedom at the node of the element. Specify mass, not weight. You must associate this mass with a region of your model.

Input File Usage:
**MASS*, ELSET=*name*
mass magnitude

where the ELSET parameter refers to a set of MASS elements.

Abaqus/CAE Usage: Property or Interaction module: **Special->Inertia->Create**: **Point mass/inertia**: select point: **Magnitude**: **Isotropic:** *mass magnitude*

Defining the Mass Matrix Explicitly in Abaqus/Standard

You can define a general mass matrix explicitly in Abaqus/Standard if the introduction of individual terms on and off the diagonal of the mass matrix is desired. See *User-Defined Elements* for details.

Input File Usage: Use both of the following options:

**USER ELEMENT*
**MATRIX*

Abaqus/CAE Usage: Defining the mass matrix explicitly is not supported in Abaqus/CAE.

Defining the Anisotropic Mass Tensor

You can specify the mass as anisotropic by giving the three principal values and the principal directions. When the orientation of the principal directions is not specified, they are assumed to coincide with the global axes. In a large-displacement analysis the local axes of the anisotropic mass rotate with the rotation, if active, of the node to which the anisotropic mass is attached. The rotation degree of freedom is active at a node if that node is connected to a beam, a conventional shell, a rotary inertia element, or a rigid body. You can specify mass proportional loads such as gravitation on an anisotropic mass. Damping and mass scaling can also be used with an anisotropic mass.

Specify mass, not weight. You must associate this mass with a region of your model.

Input File Usage: **MASS*, ELSET=*name*, TYPE=ANISOTROPIC, ORIENTATION=*orientation_name*
M_{11}, M_{22}, M_{33}

where the ELSET parameter refers to a set of MASS elements.

Abaqus/CAE Usage: Property or Interaction module: **Special->Inertia->Create**: **Point mass/inertia**: select point: **Magnitude**: **Anisotropic:** M_{11}, M_{22}, and M_{33}

Defining Damping for MASS Elements

In Abaqus/Standard you can define mass proportional damping for direct-integration dynamic analyses and mode-based analyses that support nondiagonal damping, or you can define composite damping for mode-based analyses that do not support nondiagonal damping. Although both damping definitions can be specified for a set of MASS elements, only the damping that is relevant to the particular dynamic analysis procedure will be used.

In Abaqus/Explicit mass proportional damping can be defined for MASS elements.

See *Material Damping* for details.

Input File Usage: Use the following option to define mass proportional damping:

**MASS*, ALPHA=α_R

Use the following option to define the fraction of critical damping to be used with the MASS elements when calculating composite damping factors for the modes when used in modal dynamic analysis:

**MASS*, COMPOSITE=ξ_α

Abaqus/CAE Usage: Use the following option to define mass proportional damping:

Property or Interaction module: **Special->Inertia->Create**: **Point mass/inertia**: select point: **Damping**: **Alpha:** α_R

Use the following option to define the fraction of critical damping to be used with the MASS elements when calculating composite damping factors for the modes when used in modal dynamic analysis:

Property or Interaction module: **Special->Inertia->Create**: **Point mass/inertia**: select point: **Damping**: **Composite:** ξ_α

Mass Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the mass elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *Point Masses*
- **MASS*

Element Types

MASS Point mass

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage: **MASS*

Abaqus/CAE Usage: Property or Interaction module: **Special->Inertia->Create**: **Point mass/inertia**: select point: **Magnitude**: **Mass**

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): CENTRIF(S)
Abaqus/CAE Load/Interaction: Not supported

Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Not supported
Units: LT^{-2}
Description: Gravity loading in a specified direction.

Load ID (*DLOAD): ROTA(S)
Abaqus/CAE Load/Interaction: Not supported
Units: T^{-2}
Description: Rotary acceleration load (magnitude is input as α, where α is the rotary acceleration).

Element Output

ELKE	Element kinetic energy (available only from Abaqus/Standard).

Nodes Associated with the Element

1 node.

Rotary Inertia Elements

In this section:

- *Rotary Inertia*
- *Rotary Inertia Element Library*

Rotary Inertia

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Rotary Inertia Element Library*
- **ROTARY INERTIA*
- *Defining point mass and rotary inertia*

Overview

Rotary inertia elements:

- allow rotary inertia to be included at a node;
- are associated with the three rotational degrees of freedom at a node; and
- can be paired with a MASS element (*Point Masses*) to define the mass and inertia properties of a rigid body directly (*Rigid Body Definition*).

Defining the Rotary Inertia

The ROTARYI element allows rotary inertia to be included at a node. The node is assumed to be the center of mass of the body so that only second moments of inertia are required. If the node is part of a rigid body, the offset between the node and the center of mass of the rigid body is accounted for. All six components of the rotary inertia tensor—I_{11}, I_{22}, I_{33}, I_{12}, I_{13}, and I_{23}—about the global coordinate system are defined as follows:

$$
\begin{aligned}
I_{11} &= \int_V \rho \left((x_2)^2 + (x_3)^2 \right) \, dV \\
I_{22} &= \int_V \rho \left((x_3)^2 + (x_1)^2 \right) \, dV \\
I_{33} &= \int_V \rho \left((x_1)^2 + (x_2)^2 \right) \, dV \\
I_{12} &= -\int_V \rho \left(x_1 x_2 \right) \, dV \\
I_{13} &= -\int_V \rho \left(x_1 x_3 \right) \, dV \\
I_{23} &= -\int_V \rho \left(x_2 x_3 \right) \, dV.
\end{aligned}
$$

The rotary inertia tensor must be positive semi-definite.

You specify the moments of inertia, which should be given in units of ML^2. You must associate these moments of inertia with a region of your model.

Optionally, you can refer to a local orientation (*Orientations*) that defines the directions of the local axes for which the rotary inertia values are being given. If you do not specify a local orientation and the rotary inertia element is defined within a part or a part instance (see *Assembly Definition*), the components of the inertia tensor must be given with respect to the local part axes. If you do not specify a local orientation and the rotary inertia element is not defined within a part or a part instance, the components of the inertia tensor must be given with respect to the global axes.

Input File Usage: **ROTARY INERTIA*, ELSET=*name*, ORIENTATION=*name*
I_{11}, I_{22}, I_{33}, I_{12}, I_{13}, I_{23}

where the ELSET parameter refers to a set of ROTARYI elements.

Abaqus/CAE Usage: Property or Interaction module: **Special->Inertia->Create**: **Point mass/inertia**: select point: **Magnitude**: **I11:** I_{11}, **I22:** I_{22}, **I33:** I_{33}; if necessary, toggle on **Specify off-diagonal terms**: **I12:** I_{12}, **I13:** I_{13}, **I23:** I_{23}; **CSYS: Edit**

Defining Damping for ROTARYI Elements

In Abaqus/Standard you can define mass proportional damping for direct-integration dynamic analysis or composite damping for modal dynamic analysis. Although both damping definitions can be specified for a set of ROTARYI elements, only the damping that is relevant to the particular dynamic analysis procedure will be used.

In Abaqus/Explicit mass proportional damping can be defined for ROTARYI elements.

Dynamics

You can define inertia proportional damping for ROTARYI elements in direct-integration dynamic analysis or explicit dynamic analysis. See *Material Damping* for details.

Input File Usage: **ROTARY INERTIA*, ALPHA=α_R

Abaqus/CAE Usage: Property or Interaction module: **Special->Inertia->Create**: **Point mass/inertia**: select point: **Damping**: **Alpha:** α_R

Modal Dynamics

You can define the fraction of critical damping to be used with the ROTARYI elements when calculating composite damping factors for the modes when used in modal dynamic analysis. See *Material Damping* for details.

Input File Usage: **ROTARY INERTIA*, COMPOSITE=ξ_α

Abaqus/CAE Usage: Property or Interaction module: **Special->Inertia->Create**: **Point mass/inertia**: select point: **Damping**: **Composite:** ξ_α

Speeding Up Convergence in Three-Dimensional Implicit Analyses

In geometrically nonlinear analysis in Abaqus/Standard, rigid body rotary inertia contributes some unsymmetric terms to the system matrix when the motion is in three dimensions and the rotary inertia is not the same about all three axes. Therefore, in cases when the rotary inertia effects are significant, the solution may converge faster if you use the unsymmetric matrix storage and solution scheme for the step (*Defining an Analysis*).

Rotary Inertia Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the rotary inertia elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *Rotary Inertia*
- **ROTARY INERTIA*

Element Types

ROTARYI Rotary inertia at a point

Active Degrees of Freedom

4, 5, 6

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage: **ROTARY INERTIA*

Abaqus/CAE Usage: Property or Interaction module: **Special->Inertia->Create**: **Point mass/inertia**: select point: **Magnitude**: **Rotary Inertia**

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): ROTA(S)
Abaqus/CAE Load/Interaction: Not supported

Units: T^{-2}

Description: Rotary acceleration load (magnitude is input as α, where α is the rotary acceleration).

Load ID (*DLOAD): ROTDYNF(S)

Abaqus/CAE Load/Interaction: Not supported

Units: T^{-1}

Description: Rotordynamic load (magnitude is input as ω, where ω is the angular velocity).

Element Output

ELKE Element kinetic energy (available only from Abaqus/Standard).

Nodes Associated with the Element

1 node.

Rigid Elements

In this section:

- *Rigid Elements*
- *Rigid Element Library*

Rigid Elements

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Rigid Body Definition*
- *Rigid Element Library*
- **RIGID BODY*
- *Defining rigid body constraints*

Overview

Rigid elements:

- can be used to define the surfaces of rigid bodies for contact;
- can be used to define rigid bodies for multibody dynamic simulations;
- can be attached to deformable elements;
- can be used to constrain parts of a model;
- are used to apply Abaqus/Aqua loads to rigid structures; and
- are associated with a given rigid body and share a common node known as the rigid body reference node.

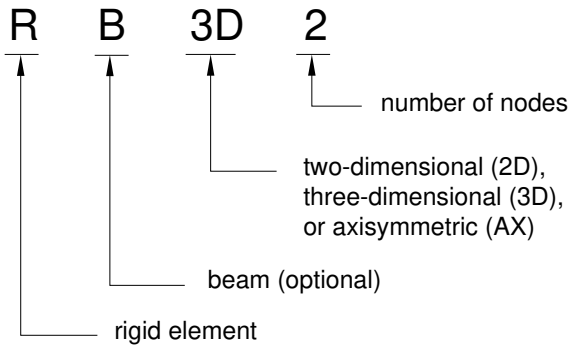
Choosing an Appropriate Element

Use R2D2 elements in plane strain or plane stress analysis, RAX2 elements in axisymmetric planar geometries, and R3D3 and R3D4 elements in three-dimensional analysis.

RB2D2 and RB3D2 elements are often used in Abaqus/Standard to model offshore structures that will transmit Abaqus/Aqua loads but will not deform. They can also be used as rigid links between nodes on deformable bodies.

Naming Convention

Rigid elements in Abaqus are named as follows:



For example, R2D2 is a two-dimensional, 2-node, rigid element.

Element Normal Definition

For all rigid elements the face on the side of the element with the positive outward normal is referred to as SPOS. The face on the opposite side is referred to as SNEG. The positive normal direction for each element is defined below.

R2D2, RAX2, RB2D2, R3D3, and R3D4 rigid elements can be used in Abaqus/Standard to define main surfaces for contact applications. The direction of the main surface's outward normal is critical for proper detection of contact. See *About Contact Pairs in Abaqus/Standard* for a more detailed discussion of contact surface definitions.

Two-Dimensional Rigid Elements

The positive outward normal direction, $\mathbf{n}$, is defined by a 90° counterclockwise rotation from the direction going from node 1 to node 2 of the element. See *Figure 1*.

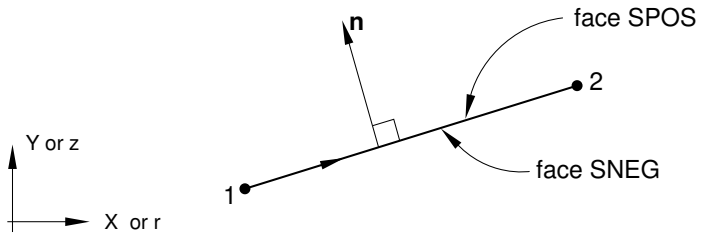


Figure 1: Positive normal for two-dimensional rigid elements.

Three-Dimensional Rigid Elements

The positive normal for R3D3 and R3D4 elements is given by the right-hand rule going around the nodes of the element in the order that they are given in the element's connectivity. See *Figure 2*.

RB3D2 elements do not have a unique normal definition.

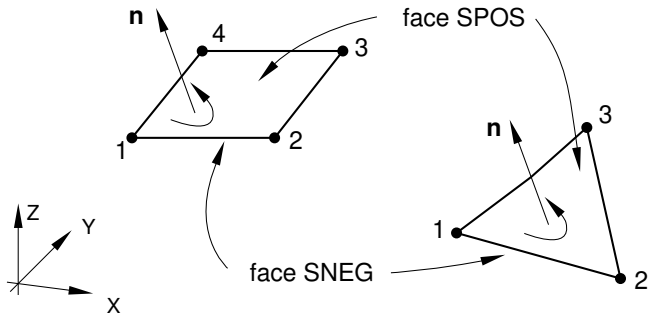


Figure 2: Positive normals for R3D3 and R3D4 elements.

Defining Rigid Elements

Rigid elements must always be part of a rigid body. See *Rigid Body Definition* for complete details on the definition of a rigid body.

Input File Usage: **RIGID BODY*, ELSET=*name*

where the ELSET parameter refers to a set of rigid elements.

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Rigid body**: **Body (elements)**

Mass Distribution

In Abaqus/Standard rigid elements do not contribute mass to the rigid body to which they are assigned. The mass distribution on the rigid surface can be accounted for by using point mass (*Point Masses*) and rotary inertia elements (*Rotary Inertia*) on the nodes connected to the rigid elements.

By default in Abaqus/Explicit, rigid elements do not contribute mass to the rigid body to which they are assigned. To define the mass distribution, you can specify the density of all rigid elements in a rigid body. When a nonzero density and thickness are specified, mass and rotary inertia contributions to the rigid body from rigid elements will be computed in an analogous manner to structural elements.

Input File Usage: Use the following option in Abaqus/Explicit to specify the density of rigid elements:

**RIGID BODY*, DENSITY=*density*

Abaqus/CAE Usage: You cannot specify the density of rigid elements in Abaqus/CAE.

Geometry in Abaqus/Explicit

In Abaqus/Explicit you can specify the cross-sectional area or thickness for all of the rigid elements that are part of a rigid body. Abaqus/Explicit assumes a default zero cross-sectional area or thickness if you do not specify one.

To account for a continuously varying thickness of a surface formed by rigid elements in Abaqus/Explicit, you can specify the thickness of the rigid elements at the nodes.

Specifying a nonzero thickness for rigid elements that form a rigid surface in a contact pair definition can be used to account for the effect of surface thickness in the contact constraint. It also enables the use of the double-sided surface contact feature with rigid surfaces formed by rigid elements.

Input File Usage: Use the following option in Abaqus/Explicit to specify the cross-sectional area or thickness for all rigid elements in a rigid body:

**RIGID BODY*
cross-sectional area or thickness

Use both of the following options to specify a continuously varying thickness for a surface formed by rigid elements:

**NODAL THICKNESS*
**RIGID BODY*, NODAL THICKNESS

Abaqus/CAE Usage: You cannot specify the cross-sectional area or thickness of rigid elements in Abaqus/CAE.

Offset in Abaqus/Explicit

In Abaqus/Explicit you can define the distance (measured as a fraction of the rigid element's thickness) from the rigid element's midsurface to the reference surface containing the element's nodes. The positive values of the offset are in the direction of the element normal. When the offset distance is 0.5, the top surface is the reference surface. When the offset distance is −0.5, the bottom surface is the reference surface. The default offset distance is 0, which indicates that the middle surface of the rigid element is the reference surface. You can specify a value for the offset distance that is greater in magnitude than half the rigid element's thickness.

Since no element-level calculations are performed for rigid elements, a specified offset affects only the handling of contact pairs with rigid surfaces formed by rigid elements (see *Element-Based Surface Definition*). Mass and rotary inertia contributions to the rigid body from rigid elements defined with an offset are computed as if the offset is zero.

Input File Usage: Use the following option in Abaqus/Explicit to specify a surface offset for a rigid element:

**RIGID BODY*, OFFSET=*offset*

The OFFSET parameter accepts a value or a label (SPOS or SNEG). Specifying SPOS is equivalent to specifying a value of 0.5; specifying SNEG is equivalent to specifying a value of −0.5.

Abaqus/CAE Usage: You cannot specify an offset for rigid elements in Abaqus/CAE.

Rigid Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the rigid elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *Rigid Elements*
- **RIGID BODY*

Element Types

2D Rigid Elements

R2D2	2-node, linear link (for use in plane strain or plane stress)
RAX2	2-node linear axisymmetric rigid link (for use in axisymmetric planar geometries)
RB2D2$^{(S)}$	2-node, rigid beam

Secondary Degrees of Freedom

R2D2 and RAX2: 1, 2

RB2D2: 1, 2, 6

Main Degrees of Freedom

R2D2, RAX2, and RB2D2: 1, 2, 6 at the rigid body reference node

Additional Solution Variables

None.

3D Rigid Elements

R3D3	3-node, triangular facet
R3D4	4-node, bilinear quadrilateral
RB3D2$^{(S)}$	2-node, rigid beam

Secondary Degrees of Freedom

R3D3 and R3D4: 1, 2, 3

RB3D2: 1, 2, 3, 4, 5, 6

Main Degrees of Freedom

1, 2, 3, 4, 5, 6 at the rigid body reference node

Additional Solution Variables

None.

Nodal Coordinates Required

R2D2 and RB2D2: *X*, *Y*

RAX2: *r*, *z*

R3D3, R3D4, and RB3D2: *X*, *Y*, *Z*

Element Property Definition

For R2D2, RB2D2, and RB3D2 elements you can specify the cross-sectional area of the element in Abaqus/Explicit. The cross-sectional area cannot be specified in Abaqus/Standard.

For RAX2, R3D3, and R3D4 elements you can specify the thickness of the element in Abaqus/Explicit. The thickness of the element cannot be specified in Abaqus/Standard.

The cross-sectional area or element thickness is used for the purpose of defining body forces, which are given in units of force per unit volume, and determining the total mass.

Input File Usage: **RIGID BODY*

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Rigid body**: **Body (elements)**

Element-Based Loading

Distributed Loads

Distributed loads are available for elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*.

Available for R2D2 elements only:

Load ID (*DLOAD): BX (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *X*-direction.

Load ID (*DLOAD): BY (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *Y*-direction.

Load ID (*DLOAD): BXNU (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}

Description: Nonuniform body force in global *X*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BYNU (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *Y*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): CENT (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-4} ($ML^{-3}T^{-2}$)

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume and ω is the angular velocity).

Load ID (*DLOAD): CORIO (S)
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-4}T$ ($ML^{-3}T^{-1}$)

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit volume and ω is the angular velocity). The load stiffness due to Coriolis loading is not accounted for in direct steady-state dynamics analysis.

Load ID (*DLOAD): P (E)
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element surface. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): PNU (E)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure on the element surface with magnitude supplied via user subroutine *VDLOAD*. The pressure is positive in the direction of the positive element normal.

Available for RAX2 elements only:

Load ID (*DLOAD): BR (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force per unit volume in the radial direction.

Load ID (*DLOAD): BZ (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force per unit volume in the axial direction.

Load ID (*DLOAD): BRNU (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force per unit volume in the radial direction, with the magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BZNU (S)

Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}

Description: Nonuniform body force per unit volume in the z-direction, with the magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): CENT (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-4} ($ML^{-3}T^{-2}$)

Description: Centrifugal load (magnitude given as $\rho\omega^2$, where ρ is the mass density and ω is the angular speed). Since only axisymmetric deformation is allowed, the spin axis must be the z-axis.

Load ID (*DLOAD): HP (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Hydrostatic pressure on the element surface and linear in global *Z*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}

Description: Pressure on the element surface. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): PNU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Nonuniform pressure on the element surface with the magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Shear traction on the element surface.

Load ID (*DLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Nonuniform shear traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: General traction on the element surface.

Load ID (*DLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Nonuniform general traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Available for R3D3 and R3D4 elements only:

Load ID (*DLOAD): BX (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in the global *X*-direction.

Load ID (*DLOAD): BY (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in the global *Y*-direction.

Load ID (*DLOAD): BZ (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in the global *Z*-direction.

Load ID (*DLOAD): BXNU (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in the global *X*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BYNU (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in the global *Y*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BZNU (S)
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in the global *Z*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): CENT (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-4} ($ML^{-3}T^{-2}$)

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume and ω is the angular velocity).

Load ID (*DLOAD): CORIO (S)
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-4}T$ ($ML^{-3}T^{-1}$)

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit volume and ω is the angular velocity). The load stiffness due to Coriolis loading is not accounted for in direct steady-state dynamics analysis.

Load ID (*DLOAD): HP (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Hydrostatic pressure on the element surface and linear in global *Z*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element surface. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): PNU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure on the element surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element surface.

Load ID (*DLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform shear traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element surface.

Load ID (*DLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform general traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Abaqus/Aqua Loads

Abaqus/Aqua loads are specified as described in *Abaqus/Aqua Analysis*.

Available for R3D3 and R3D4 elements only:

Load ID (*CLOAD/ *DLOAD): PB
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Buoyancy force.

Available for RB2D2 and RB3D2 elements only:

Load ID (*CLOAD/ *DLOAD): FDD
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Transverse fluid drag force.

Load ID (*CLOAD/ *DLOAD): FD1

Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Fluid drag force on the first end of the rigid link (node 1).

Load ID (*CLOAD/ *DLOAD): FD2
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Fluid drag force on the second end of the rigid link (node 2).

Load ID (*CLOAD/ *DLOAD): FDT
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Tangential fluid drag load.

Load ID (*CLOAD/ *DLOAD): FI
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Transverse fluid inertia load.

Load ID (*CLOAD/ *DLOAD): FI1
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Fluid inertia load on the first end of the rigid link (node 1).

Load ID (*CLOAD/ *DLOAD): FI2
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Fluid inertia load on the second end of the rigid link (node 2).

Load ID (*CLOAD/ *DLOAD): PB
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Buoyancy force (with closed-end condition).

Load ID (*CLOAD/ *DLOAD): WDD
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Transverse wind drag force.

Load ID (*CLOAD/ *DLOAD): WD1
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Wind drag force on the first end of the rigid link (node 1).

Load ID (*CLOAD/ *DLOAD): WD2
Abaqus/CAE Load/Interaction: Not supported
Units: F
Description: Wind drag force on the second end of the rigid link (node 2).

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are available for elements with displacement degrees of freedom. They are specified as described in *Distributed Loads*.

Available for RAX2, R3D3, and R3D4 elements only:

Load ID (*DSLOAD): HP (S)
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Hydrostatic pressure on the element surface and linear in global *Z*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element surface. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Nonuniform pressure on the element surface with the magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element surface.

Load ID (*DSLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform shear traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element surface.

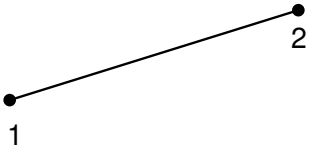
Load ID (*DSLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform general traction on the element surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Element Output

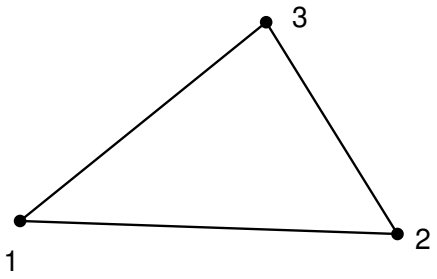
None.

Node Ordering on Elements

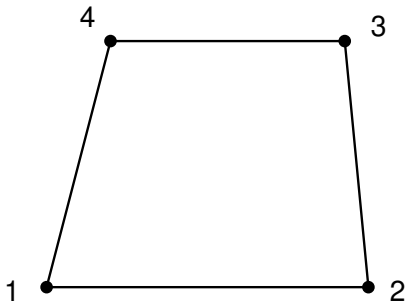
R2D2, RAX2
RB2D2, RB3D2



R3D3



R3D4



Capacitance Elements

In this section:

- *Point Capacitance*
- *Capacitance Element Library*

Point Capacitance

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Capacitance Element Library*
- **HEATCAP*
- *Defining heat capacitance*

Overview

Capacitance elements:

- allow the introduction of concentrated heat capacitance at a point;
- are associated with the temperature degree of freedom at a node; and
- have a capacitance that can be specified as a function of temperature and/or field variables.

Defining the Capacitance Value

The heat capacitance is associated with the temperature degree of freedom at the node of the element. You specify the capacitance magnitude, ρcV (density × specific heat × volume). Specify capacitance, not specific heat. You must associate this capacitance with a region of your model.

Input File Usage: **HEATCAP*, ELSET=*name*
ρcV

where the ELSET parameter refers to a set of HEATCAP elements.

Abaqus/CAE Usage: Property or Interaction module: **Special->Inertia->Create**: **Heat capacitance**: select points: **Capacitance** ρcV

Capacitance Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the capacitance elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *Point Capacitance*
- **HEATCAP*

Element Types

HEATCAP Point heat capacitance

Active Degrees of Freedom

11

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage: **HEATCAP*

Abaqus/CAE Usage: Property or Interaction module: **Special->Inertia->Create**: **Heat capacitance**

Element-Based Loading

None.

Element Output

None.

Nodes Associated with the Element

1 node.

Connector Elements

In this section:

- *Connector Elements*
- *Connector Element Behavior*

Connector Elements

In this section:

- *About Connectors*
- *Connector Elements*
- *Connector Actuation*
- *Connector Element Library*
- *Connection Types*

About Connectors

Overview

Connector modeling consists of:

- choosing and defining the appropriate connector elements (*Connector Elements*);
- defining the connector behavior (*Connector Behavior*);
- defining any connector actuations (*Connector Actuation*); and
- monitoring connector output (*Connector Elements* and *Connector Element Library*).

Abaqus offers a library of connector types and connector elements to model the behavior of connectors.

References:

Typical Applications

The analyst is often faced with modeling problems in which two different parts are connected in some way. Sometimes connections are simple, such as two panels of sheet metal spot welded together or a door connected to a frame with a hinge. In other cases the connection may impose more complicated kinematic constraints, such as constant velocity joints, which transmit constant spinning velocity between misaligned and moving shafts. In addition to imposing kinematic constraints, connections may include (nonlinear) force versus displacement (or velocity) behavior in their unconstrained relative motion components, such as a muscle force resisting the rotation of a knee joint in a crash-test occupant model. More complex connections may include the following:

- stopping mechanisms, which restrict the range of motion of an otherwise unconstrained relative motion;
- internal friction, such as the lateral force or moments on a bolt generating friction in the translation of the bolt along a slot;
- failure conditions, where excess force or displacement inside the connection causes the entire connection or a single component of relative motion to break free; and
- locking mechanisms that engage after some force or displacement criteria is met, such as a snap-fit connector or a falling-pin locking mechanism on a satellite deployment arm.

In many situations the connection can be actuated either through displacement or force control, such as a hydraulic piston or a gear-driven robot arm.

In Abaqus/Standard if the two parts being connected are rigid bodies, multi-point constraints cannot be used to connect the bodies at nodes other than the reference nodes, since multi-point constraints use degree-of-freedom elimination and the other nodes on a rigid body do not have independent degrees of freedom. In Abaqus/Explicit this restriction does not apply. See *General Multi-Point Constraints*.

Connector elements in Abaqus provide an easy and versatile way to model these and many other types of physical mechanisms whose geometry is discrete (i.e., node-to-node), yet the kinematic and kinetic relationships describing the connection are complex.

Connector Elements Versus Multi-Point Constraints

In many instances connector elements perform functions similar to multi-point constraints (*General Multi-Point Constraints*). However, in most cases multi-point constraints eliminate degrees of freedom at one of the nodes involved in the connection. This elimination has the advantage that the problem size is reduced; it has the disadvantage that output and other functionality provided with connector elements is not available. In addition, in Abaqus/Standard the degree of freedom elimination prevents the use of multi-point constraints between nodes without independent degrees of freedom (such as nodes on a rigid body whose degrees of freedom are dependent on the degrees of freedom at the reference node).

In contrast, connector elements do not eliminate degrees of freedom; kinematic constraints are enforced with Lagrange multipliers. These Lagrange multipliers are additional solution variables in Abaqus/Standard. The Lagrange multipliers provide constraint force and moment output. Since connector elements do not eliminate degrees of freedom, they can be used in many situations where multi-point constraints cannot be used or do not exist for the function required; for example, to connect two rigid bodies at nodes other than the reference node in Abaqus/Standard.

Multi-point constraints are more efficient than connector elements; and if the requirements of the analysis can be satisfied with multi-point constraints, they should be used in place of connector elements.

Input File Template

The following template shows the options used to define and activate the connector elements shown in *Figure 1* and *Figure 2*. In the respective figures on the left is a schematic representation of a connection to be modeled; on the right is a representation of the equivalent finite element model. All options are discussed in detail in the following sections.

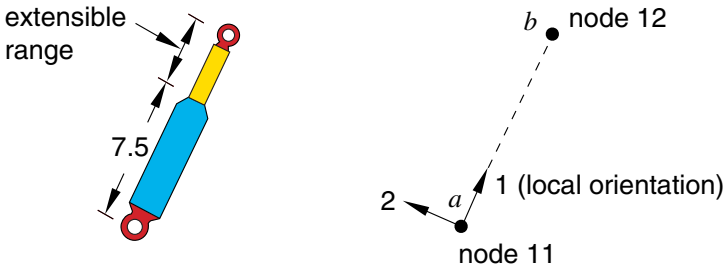


Figure 1: Simplified connector model of a shock absorber.

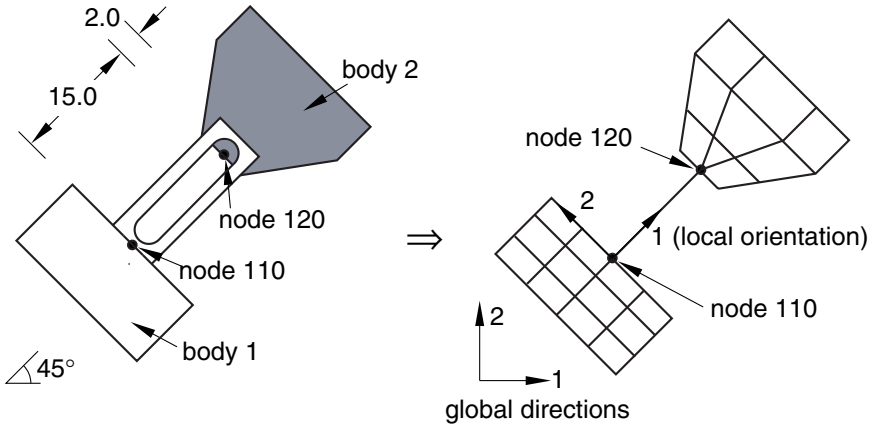


Figure 2: A pin-in-slot connection modeled with SLOT and CARDAN connection types.

```
*HEADING
…
*ELEMENT, TYPE=CONN3D2, ELSET=shock
101, 11, 12
*ELEMENT, TYPE=CONN3D2, ELSET=pininslot
1010, 110, 120
…
*ORIENTATION, NAME=ori60
0.5, 0.866025, 0.0, -0.866025, 0.5, 0.0
*ORIENTATION, NAME=ori45
0.707, 0.707, 0.0, -0.707, 0.707, 0.0
*CONNECTOR SECTION, ELSET=shock, BEHAVIOR=sbehavior
revolute, slot
ori60,
…
*CONNECTOR BEHAVIOR, NAME=sbehavior
*CONNECTOR DAMPING, COMPONENT=1
1500.0

*CONNECTOR LOCK, COMPONENT=3, LOCK=4
 , , -500.0, 500.0
*CONNECTOR ELASTICITY, COMPONENT=4, NONLINEAR
-900., -0.7
   0.,  0.0
1250.,  0.7
*CONNECTOR CONSTITUTIVE REFERENCE
 , , , 22.5,
*CONNECTOR STOP, COMPONENT=1
7.5, 15.0
…
*CONNECTOR FRICTION
0.34, 0.55,  0.0
0.34, 0.10,  0.45
*FRICTION
.15
…
*CONNECTOR SECTION, ELSET=pininslot
cardan, slot
ori45,
*CONNECTOR MOTION
pininslot, 4
pininslot, 5
```

```
…
*STEP
…
*CONNECTOR MOTION, TYPE=VELOCITY
pininslot, 6, 0.7854
…
*CONNECTOR LOAD
pininslot, 1, 1000.0
…
*END STEP
```

Connector Elements

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Connectors*
- *Connector Element Library*
- *Connection-Type Library*
- *Connection Types*
- **CONNECTOR SECTION*
- *Creating connector sections*
- *Creating and modifying connector section assignments*

Overview

Connector elements:

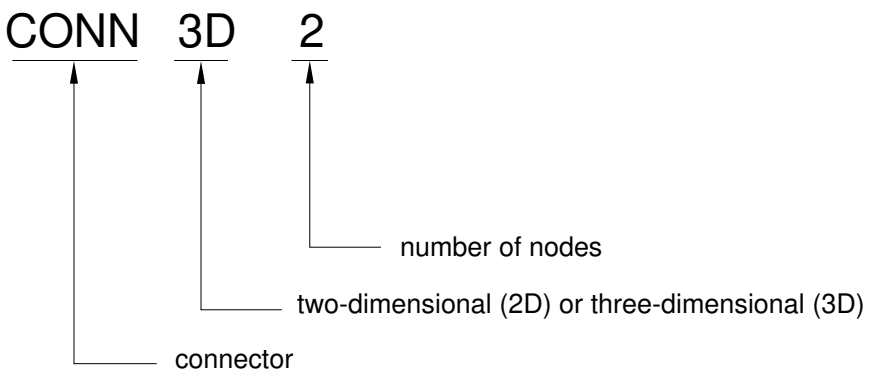
- are available for two-dimensional, axisymmetric, and three-dimensional analyses;
- can define a connection between two nodes (each node can be connected to a rigid part, a deformable part, or not connected to any part);
- can define a connection between a node and ground;
- have relative displacements and rotations that are local to the element, which are referred to as components of relative motion;
- are functionally defined by specifying the connector attributes;
- have comprehensive kinematic and kinetic output; and
- can be used to monitor kinematics in local coordinate systems.

Choosing an Appropriate Element

Two connector elements are provided. The element type to be chosen depends on the dimensionality of the analysis: CONN2D2 for two-dimensional and axisymmetric analyses and CONN3D2 for three-dimensional analyses. Both connector elements have at most two nodes. The position and motion of the second node on the connector element are measured relative to the first node.

Naming Convention

Connector elements in Abaqus are named as follows:



For example, CONN2D2 is a two-dimensional, 2-node connector element.

Defining a Connection between Points

A connector element can be used to connect two points.

Input File Usage: **ELEMENT*, TYPE=*name*
connector element number, *node_1*, *node_2*

Abaqus/CAE Usage: Interaction module: **Connector**->**Assignment**->**Create**: select wires

Defining a Connection between a Point and Ground

A connector element can be connected to ground, and the ground "node" can be the first or second point on the connector element. The initial position of the ground node used for calculating relative position and displacement is the initial position of the other point on the element. All displacements and rotations at the ground node, if they exist, are fixed.

Input File Usage: Use one of the following options:

**ELEMENT*, TYPE=*name*
connector element number, node number on the body
**ELEMENT*, TYPE=*name*
connector element number, , node number on the body

Abaqus/CAE Usage: Interaction module: **Connector**->**Assignment**->**Create**: select wires connected to ground

Components of Relative Motion

Connector elements have relative displacements and rotations that are local to the element. These relative displacements and rotations are referred to as components of relative motion. In the three-dimensional case connector elements use 12 nodal degrees of freedom to define six relative motion components: three displacements and three rotations in element local directions. In two dimensions six nodal degrees of freedom define three relative motion components: two displacements and one rotation. The components of relative motion are either constrained or unconstrained ("available"), depending on the definition of the connector element.

Constrained Components of Relative Motion

Constrained components of relative motion are displacements and rotations that are fixed by the connector element.

In connector elements with constrained components of relative motion, Abaqus/Standard uses Lagrange multipliers to enforce the kinematic constraints. Accordingly, in Abaqus/Standard the constraint forces and moments carried by the element appear as additional solution variables. The number of additional solution variables is equal to the number of constrained components of relative motion. In Abaqus/Explicit the constraints are enforced using an augmented Lagrangian technique for which no additional solution variables are needed.

Available Components of Relative Motion

Available components of relative motion are displacements and rotations that are not constrained kinematically and, hence, remain available for defining material-like behavior, specifying time-dependent motion, applying loading, or assigning complex interactions, such as contact or friction. Many connection types have available components of relative motion, and their meaning is described in *Connection-Type Library* for each individual connection type.

Defining the Connection Attributes

The connection attributes define the connector element's function. In the most general case you specify the following attributes:

- the connection type or types,
- the local directions associated with the connector's nodes,
- additional data for certain connection types, and
- the connector behavior.

The connector definition that is defined with these attributes is associated with a set of connector elements.

Input File Usage: **CONNECTOR SECTION*, ELSET=*name*

Abaqus/CAE Usage: Interaction module:
Connector->Geometry->Create Wire Feature
Connector->Section->Create: **Name:** *connector section name*
Connector->Assignment->Create: select wires: **Section:** *connector section name*

Defining the Connection Type

Abaqus provides a comprehensive library of connection types. See *Connection Types* for the available connection types. The connection types are divided into three categories: basic connection components, assembled connections, and complex connections. The basic connection components affect either translations or rotations on the second node. A connector element might include one translational basic connection component and/or one rotational basic connection component. The assembled connections are constructed from the basic connection components. They are provided for convenience and cannot be combined in the same connector element definition with a basic connection component or other assembled connections. Complex connections affect a combination of degrees of freedom at the nodes in the connection and cannot be combined with other connection components.

The connection type is specified as:

- a single basic connection type (translational or rotational),
- one translational and one rotational basic connection type,
- one assembled connection type, or

- one complex connection type.

Input File Usage: Use one of the following options:

**CONNECTOR SECTION*, ELSET=*name*
basic connection type, *basic connection type*
**CONNECTOR SECTION*, ELSET=*name*
assembled connection or *complex connection*

Abaqus/CAE Usage: Interaction module:
Connector->Section->Create: **Connection Category**: **Basic**, **Translational type:** *translational basic connection type* and/or **Rotational type:** *rotational basic connection type*
or
Connector->Section->Create: **Connection Category**: **Assembled/Complex**, **Assembled/Complex type:** *assembled connection* or *complex connection*

Defining the Local Connector Directions

Local directions at the nodes are often required to define the connection types used to define the connector element. The local directions and how they are used to define the connection are described in *Connection Types*. In the most general case the connection type uses two sets of local directions, which are defined as described in *Orientations*. The names associated with the two orientation definitions must be referred to from the connector section definition. Associated orientations can be defined as distributions, and the names of the distributions must be referred to from the connector section definitions.

Input File Usage: Use the following options for the most general case:

**ORIENTATION*, NAME=*orientation_1*
**ORIENTATION*, NAME=*orientation_2*
**CONNECTOR SECTION*, ELSET=*name*
basic connection type(s) or *assembled connection*
orientation_1 for first node (or ground) or the name of the orientation distribution,
orientation_2 for second node (or ground) or the name of the orientation distribution

Abaqus/CAE Usage: Interaction module: **Connector->Assignment->Create**: select wires: **Orientation 1**, **Orientation 2**: **Edit**: select the orientations for the first and second points, respectively, of the selected wires

Degree of Freedom Activation and Corotation of Connection Directions

Many connection types either require connection directions at the nodes on the element or allow optional directions to be defined. In cases where an orientation definition is permitted for defining connection directions (required or optional), connector elements activate the rotational degrees of freedom at the nodes to which they are attached, if they do not exist already. Orientation definitions are allowed for the following basic translational connection types: ACCELEROMETER, CARTESIAN, PROJECTION CARTESIAN, RADIAL-THRUST, SLIDE-PLANE, and SLOT. Consequently, rotational degrees of freedom will be activated, if they do not exist already, when any of these basic translational connection types are used. The only exception is connection type JOIN, for which connection directions are optional at the first node of the element, but rotation degrees of freedom are not activated. Only two basic translational connection types, LINK and AXIAL, will ignore the specified orientation

definition at both node *a* and node *b*. Connector elements using one of these two connection types will not activate rotational degrees of freedom, if they do not exist already.

The connector element's orientation directions corotate with the rotational degrees of freedom at the corresponding node on the element. If there is no element with rotational degrees of freedom or rotation constraint (such as an equation or a multi-point constraint) attached to the node, you must ensure that sufficient rotational boundary conditions are provided to avoid numerical singularities associated with unconstrained rotational degrees of freedom. Connection type JOIN uses fixed directions when rotational degrees of freedom are not active at the nodes on the connector element.

Example

Figure 1 illustrates the use of the CONN3D2 element to connect two bodies with a cylindrical-like connector oriented at 60° from the global 1-axis. On the left is a schematic representation of the connection to be modeled; on the right is a representation of the equivalent finite element model. See *Connection-Type Library* for a list of connector type names.

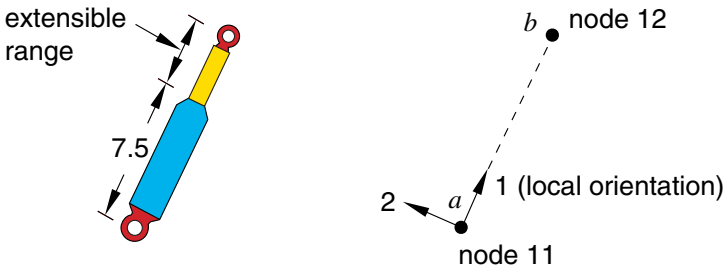


Figure 1: Simplified connector model of a shock absorber.

The connection requires node *b* to remain on the line of the shock absorber, which is determined by the position and orientation directions of node *a*. In addition, the two rotation components perpendicular to the line of the shock absorber at node *b* must be the same as those at node *a*. Hence, the only relative motion components permitted in the connection are the displacement of node *b* relative to node *a* along the line of the shock absorber and the rotation of node *b* relative to node *a* about the line of the shock absorber. This displacement and this rotation are the available components of relative motion. The connector is defined using the following lines in the input file:

```
*ELEMENT, TYPE=CONN3D2, ELSET=shock
101, 11, 12
*CONNECTOR SECTION, ELSET=shock
slot, revolute
ori60,
*ORIENTATION, NAME=ori60
**Defines the local 1-direction along the slot (required)
**Also defines the rotation axis for the revolute (required)
0.5, 0.866025, 0.0, -0.866025, 0.5, 0.0
```

Alternatively, you could use the assembled connection type CYLINDRICAL instead of the two basic connection types SLOT and REVOLUTE.

Defining Additional Connection Type Data

Some connection types allow additional data to define the kinematic behavior of the connector. For example, the connection type FLOW-CONVERTER allows you to specify a scaling factor for material flow at node *b*. See *Connection-Type Library* for more information.

Defining the Connector Behavior

Abaqus provides comprehensive kinetic behavior modeling in the available components of relative motion. Defining connector behavior is optional and can be used to incorporate spring, dashpot, node-to-node contact, locking, friction, plasticity-like effects, and failure. The kinetic modeling capabilities in connectors are described in detail in *Connector Behavior*.

Using Connector Elements in Two-Dimensional and Axisymmetric Analysis

Not all connection types can be used with element type CONN2D2. The connection-type library contains many connection types whose mechanics are valid for three-dimensional analyses only. In other cases the local directions required in the definition of the connection type conflict with the two-dimensional coordinate system. See *Connection-Type Library* for more information.

Using Multiple Connector Elements in Parallel

Connector elements in Abaqus allow most physical connections to be modeled with a single connector element. However, in certain circumstances more complex connections or output considerations might require multiple connector elements to be used in parallel. This is accomplished by defining two or more connector elements between the same nodes. In this case you must ensure that a constrained component of relative motion in one connector element is not constrained (either by a kinematic constraint or through motion specified as described in *Connector Actuation*) by one of the other connector elements.

Multiple connector elements are sometimes used in parallel to obtain output in different coordinate systems. For a connector element between two bodies, the local directions at the nodes can be determined by the requirements of the connection type. However, output might be needed in a different, possibly corotating, coordinate system. For example, the angular acceleration history could be reported in a local, body-fixed coordinate system (other than the one used to define the connector element) by using a second connector element (such as connection type CARDAN) that does not impose kinematic constraints or use connector behavior but aligns with the desired local output directions.

Defining Connectors in a Model That Contains Parts and an Assembly

An Abaqus model can be defined in terms of an assembly of part instances (see *Assembly Definition*). Connector elements can be defined at either the part level or the assembly level in such a model.

Using Connector Elements with Nodal Transformations

Nodal transformations (see *Transformed Coordinate Systems*) can be defined for either node connected to the connector element. Since these transformations affect only the nodal degrees of freedom, their use does not affect the behavior of the connector element. Connector elements operate on components of motion local to the connection.

Using Nonlinear Connections in Geometrically Linear Analyses

If a connector element with a nonlinear kinematic constraint is used in a geometrically linear analysis, the kinematic constraint is linearized. For example, if connection type LINK is used in a geometrically linear analysis, the distance between the two nodes is held constant after projection onto the direction of the line between the original positions of the nodes. The difference should be noticeable only if the magnitudes of the rotations and displacements are not small.

Mismatched Masses at Connector Nodes in Abaqus/Explicit

If the nodes of a connector element in Abaqus/Explicit have masses that are highly mismatched, the implicit solver might encounter convergence problems due to the resulting ill-conditioned coefficient matrix. To prevent this from happening, if the nodal masses or rotary inertias of a connector element differ by more than three orders

of magnitude, Abaqus/Explicit adds mass/rotary inertia to the connector element node that has the smaller mass/rotary inertia. The mass/rotary inertia added is negligibly small (less than three orders of magnitude smaller) compared to the larger of the connector element's nodal inertias. This additional mass almost never affects the solution significantly. However, in certain situations (for example, for a strongly dynamic analysis that has connector elements with highly mismatched nodal masses) this adjustment might have a noticeable effect.

Connector Output

The connector element force, moment, and kinematic output is defined in *Connector Element Library*. These output quantities include total, elastic, viscous, and friction forces and moments. In addition, reaction forces and moments caused by connector stops and locks are available as well as connector contact forces used for friction calculation.

To obtain accurate reaction force and moment output for connectors from Abaqus/Explicit, it might sometimes be necessary to run the analysis in double precision. In such situations a double precision run will also yield a better estimate of the work done by the reaction forces and moments, thereby providing a more accurate value of the energy due to the external work reported by Abaqus/Explicit.

Kinematic output includes relative position, relative displacement, relative velocity, relative acceleration, frictional slip, and constitutive displacements (the displacement used in the elastic force and hysteretic friction calculations defined as the difference between the current relative positions and the reference positions; see *Defining Reference Lengths and Angles for Constitutive Response*). For relative rotations the Abaqus convention of reporting angles between -2π and 2π radians is not used with connector elements. Connector element output of angles and rotational components or relative motion includes accumulated multiple rotations whose magnitudes can be arbitrarily large. Energy output is available, as are output flags to identify whether a connector has failed (in Abaqus/Explicit only), locked, or reached a connector stop.

In modal analysis procedures, element output is allowed only for the AXIAL, BUSHING, CARDAN, CARTESIAN, and ROTATION connection types when there is no connector motion associated with them.

In a geometrically linear step in Abaqus/Standard the relative position output variable does not change (in the same fashion that the nodal coordinates are output). Therefore, care must be exercised in interpreting output for connector stops and locks since they use updated coordinates.

In implicit dynamic problems, the output of some connector forces and moments can be noisy, especially at the beginning of dynamic steps.

Connector element names displayed in the output database have a "G" added at the end to indicate that this connector is connected to ground.

Using Connector Elements for Output Only

Connector elements defined without kinematic constraints or constitutive behavior can be used to monitor kinematic output in local coordinate systems. Quantities of interest include relative position, displacement, velocity, and acceleration in local coordinate parametrization. Finite rotation parameterizations include Euler and Cardan angles, rotation vector, and flexion-torsion-sweep. For an example that uses a connector element to monitor Euler angles, see *Motion of a rigid body in Abaqus/Standard*.

In Abaqus/Explicit all such connectors are solved without invoking the implicit solver, which leads to better performance in domain parallel mode (particularly when such connectors nodes overlap with other constraints such as secondary nodes of tie constraints).

Connector Actuation

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Connectors*
- **CONNECTOR LOAD*
- **CONNECTOR MOTION*
- *Defining a connector force*
- *Defining a connector moment*
- *Defining a connector displacement boundary condition*
- *Defining a connector velocity boundary condition*
- *Defining a connector acceleration boundary condition*

Overview

Connector actuation:

- is meant to model situations, such as deployment maneuvers, where a motor attached to the body loads the connection with an internal force or moment history or a hydraulic system imposes a known motion;
- can be used to fix available components of relative motion; and
- consists of driving an available component of relative motion by a prescribed displacement (rotation) or by a specified force (moment).

The prescribed relative motions and loads are in the local directions associated with the available components of relative motion for the connector.

Prescribing displacements/rotations for available components of relative motion that also include connector stop or connector lock behaviors may lead to overconstraints. Abaqus will issue a warning message if an overconstraint occurs.

Fixing Available Components of Relative Motion

A common practice is to fix available components of motion. Such fixed motion conditions can be used to customize connection types for specific applications. As an example, the REVOLUTE connection type uses the local 1-direction as the shared revolute axis and, hence, the available component of relative motion. If, for convenience, a revolute connection about the local 3-direction were needed, you could fix the relative rotations about the local 1- and 2-directions in a CARDAN connection type. In doing so, a connection type identical to the REVOLUTE connection type would be created; however, the shared axis would be the local 3-direction instead of the local 1-direction.

An example is provided later in this section in which the pin part of a pin-in-slot connection is modeled with a CARDAN connection type with fixed rotations.

Input File Usage: Use the following option in the model portion of the input file to fix available connector components of relative motion:

**CONNECTOR MOTION*

Abaqus/CAE Usage: Load module: **Create Boundary Condition**: **Step: Initial**: **Mechanical**: **Connector displacement**

Displacement-Controlled Actuation

You can specify a relative displacement, velocity, or acceleration between two parts in the connector's local directions in a manner similar to defining a boundary condition (see *Boundary Conditions*). You specify the connector element set name or connector element number; the component number identifying the available component of relative motion being actuated; and the value of the relative displacement, velocity, or acceleration.

The penalty used for enforcing connector motion may lead to a noisy solution, particularly in single precision for some models. Use of double precision is, therefore, preferable in such situations. If performance is a concern for double precision, you can run the constraint packaging and constraint solver in double precision (see *Abaqus/Standard and Abaqus/Explicit Execution*).

You cannot specify the motion of connectors in a subspace dynamic analysis.

You cannot prescribe relative motions and loads on a connector element with the complex connection type SLIPRING in Abaqus/Standard.

Input File Usage: Use the following option in the history portion of the input file to specify a relative displacement for a connector:

**CONNECTOR MOTION*, AMPLITUDE=*name*, OP=MOD *or* NEW, TYPE=DISPLACEMENT

Use the following option in the history portion of the input file to specify a relative velocity for a connector:

**CONNECTOR MOTION*, AMPLITUDE=*name*, OP=MOD *or* NEW, TYPE=VELOCITY

Use the following option in the history portion of the input file to specify a relative acceleration for a connector:

**CONNECTOR MOTION*, AMPLITUDE=*name*, OP=MOD *or* NEW, TYPE=ACCELERATION

Abaqus/CAE Usage: Load module: **Create Boundary Condition**: **Mechanical**: **Connector displacement**, **Connector velocity**, or **Connector acceleration**

Example

Figure 1 illustrates a pin-in-slot connection oriented at 45° from the global 1-axis modeled with element type CONN3D2.

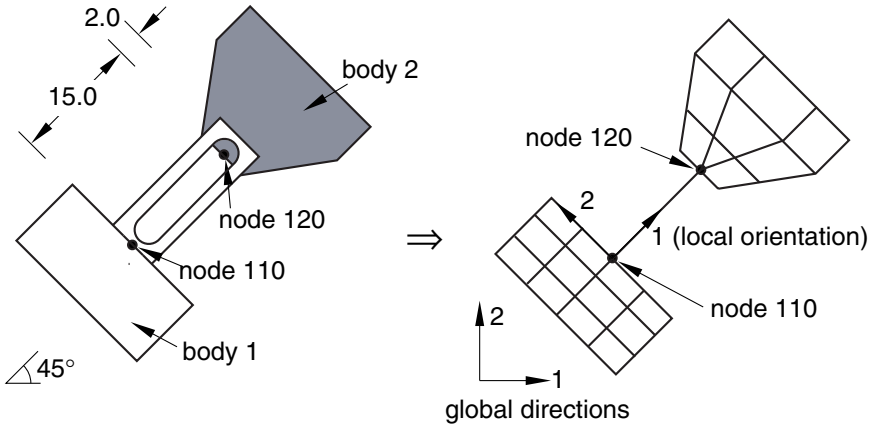


Figure 1: A pin-in-slot connection modeled with SLOT and CARDAN connection types.

The figure on the left is a schematic representation of the connection to be modeled, while the figure on the right is the finite element mesh. Displacements in the slot are allowed only along the line of the slot, and connection type SLOT is appropriate for enforcing these kinematics. Assume the pin and slot are constructed in such a way that the only rotation of the pin relative to the slot is along the local 3-direction. This is a revolute constraint; however, basic rotation connection type REVOLUTE uses the local 1-direction as the revolute axis. In this case connection type CARDAN combined with a specified constraint can be used to define a revolute-type connection with the appropriate revolute axis.

For illustrative purposes assume the connection is actuated by a rotational velocity of $\pi/4$ radians per second around the pin's axis. Using input parametrization for convenience, the following lines are used:

```
*PARAMETER
PI = 3.141592
rotangvel = PI/4
...
*ELEMENT, TYPE=CONN3D2, ELSET=pininslot
101, 110, 120
*CONNECTOR SECTION, ELSET=pininslot
cardan, slot
ori45,
*CONNECTOR MOTION
pininslot, 4
pininslot, 5
*ORIENTATION, NAME=ori45
0.707, 0.707, 0.0, -0.707, 0.707, 0.0
...
*STEP
...
*CONNECTOR MOTION, TYPE=VELOCITY
pininslot, 6, <rotangvel>
...
*END STEP
```

Force-Controlled Actuation

You can specify concentrated loads applied to the available components of relative motion in a manner similar to defining concentrated loads for other elements in Abaqus (see *Concentrated Loads*). However, connector loads are always follower loads that rotate with the rotation of the available components of relative motion as the connector element moves. You specify the connector element set name or connector element number, the component number identifying the available component of relative motion being loaded, and the value of the actuation force or moment.

Input File Usage: Use the following option in the history portion of the input file to specify a concentrated load for a connector:

**CONNECTOR LOAD*, AMPLITUDE=*name*, OP=MOD

Abaqus/CAE Usage: Load module: **Create Load**: **Mechanical**: **Connector force** or **Connector moment**

Example

Returning to the example in *Figure 1*, assume that the pin is pushed along the slot by a constant force of 1000.0 units (for example, through a hydraulic system). The following lines should be added to the input file:

```
*STEP
...
*CONNECTOR LOAD
pininslot, 1, 1000.0
...
*END STEP
```

Connector Actuation in Linear Perturbation Procedures

Nonzero magnitude connector motions are allowed only in the eigenvalue buckling, direct-solution steady-state dynamic, and linear static perturbation procedures. Any nonzero magnitude specified during an eigenfrequency extraction procedure is ignored, and the specified available component of relative motion is held fixed. Connector motions cannot be used in any modal-based procedure.

In direct-solution steady-state dynamic analyses the real and imaginary parts of any available connector component of relative motion are either restrained or unrestrained simultaneously; it is physically impossible to have one part restrained and the other part unrestrained. Abaqus/Standard will automatically restrain both the real and the imaginary parts of a component of relative motion even when only one part is prescribed specifically. The unspecified part will be assumed to have a perturbation magnitude of zero.

A nonzero prescribed connector motion in an eigenvalue buckling step will contribute to the incremental stress and, thus, will contribute to the differential initial stress stiffness. When prescribing nonzero connector motions, you must interpret the resulting eigenproblem carefully. See the discussion for boundary conditions in *Eigenvalue Buckling Prediction* for more details.

In steady-state dynamic analyses both real and imaginary connector loads can be applied in a manner similar to concentrated loads (see *Mode-Based Steady-State Dynamic Analysis*, *Direct-Solution Steady-State Dynamic Analysis*, and *Subspace-Based Steady-State Dynamic Analysis*). Multiple connector load cases can be defined in random response analyses (see *Random Response Analysis*) in the same manner as concentrated loads. Connector loads are ignored during an eigenfrequency extraction analysis.

Connector Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the connector elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *Connector Elements*
- *Connection Types*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR LOAD*
- **CONNECTOR SECTION*

Element Types

Connector in a Plane

CONN2D2 Connector element between two nodes or ground and a node.

Active Degrees of Freedom

1, 2, 6 for the most general connection types (see *Degree of Freedom Activation and Corotation of Connection Directions*).

Additional Solution Variables

In Abaqus/Standard there can be up to three additional constraint variables related to forces and a moment associated with the connector. The number of additional constraint variables depends on the connection type.

Connector in Space

CONN3D2 Connector element between two nodes or ground and a node.

Active Degrees of Freedom

1, 2, 3, 4, 5, 6 for the most general connection types (see *Degree of Freedom Activation and Corotation of Connection Directions*).

Additional Solution Variables

In Abaqus/Standard there can be up to six additional constraint variables related to forces and moments associated with the connector. The number of additional constraint variables depends on the connection type.

Nodal Coordinates Required

CONN2D2: *X*, *Y*

CONN3D2: *X*, *Y*, *Z*

Element Property Definition

Input File Usage: **CONNECTOR SECTION*

Abaqus/CAE Usage: Interaction module: **Connector->Section->Create**

Element-Based Loading

Use connector loads to apply loading to the available components of relative motion. Prescribe connector motion to specify relative kinematics (zero or nonzero values) for the available components of relative motion. See *Connector Actuation* for details.

Element Output

Total Force Components

CTF1 Total force in the 1-direction.

CTF2 Total force in the 2-direction.

CTF3 Total force in the 3-direction.

CTM1 Total moment about the 1-direction.

CTM2 Total moment about the 2-direction.

CTM3 Total moment about the 3-direction.

The total force is obtained as CTF = CEF + CVF + CUF + CSF + CRF – CCF.

Elastic Force Components

CEF1 Elastic force in the 1-direction.

CEF2 Elastic force in the 2-direction.

CEF3 Elastic force in the 3-direction.

CEM1 Elastic moment about the 1-direction.

CEM2 Elastic moment about the 2-direction.

CEM3 Elastic moment about the 3-direction.

Elastic Relative Displacement Components

CUE1 Elastic displacement in the 1-direction.

CUE2 Elastic displacement in the 2-direction.

CUE3 Elastic displacement in the 3-direction.

CURE1 Elastic rotation about the 1-direction.

CURE2 Elastic rotation about the 2-direction.

CURE3 Elastic rotation about the 3-direction.

Plastic Relative Displacement Components

CUP1 Plastic relative displacement in the 1-direction.

CUP2 Plastic relative displacement in the 2-direction.

CUP3 Plastic relative displacement in the 3-direction.

CURP1 Plastic relative rotation about the 1-direction.

CURP2 Plastic relative rotation about the 2-direction.

CURP3 Plastic relative rotation about the 3-direction.

Equivalent Plastic Relative Displacement Components

CUPEQ1 Equivalent plastic relative displacement in the 1-direction.

CUPEQ2 Equivalent plastic relative displacement in the 2-direction.

CUPEQ3 Equivalent plastic relative displacement in the 3-direction.

CURPEQ1 Equivalent plastic relative rotation about the 1-direction.

CURPEQ2 Equivalent plastic relative rotation about the 2-direction.

CURPEQ3 Equivalent plastic relative rotation about the 3-direction.

CUPEQC Equivalent plastic relative motion for a coupled plasticity definition.

Kinematic Hardening Shift Force Components

CALPHAF1 Kinematic hardening shift force in the 1-direction.

CALPHAF2 Kinematic hardening shift force in the 2-direction.

CALPHAF3 Kinematic hardening shift force in the 3-direction.

CALPHAM1 Kinematic hardening shift moment about the 1-direction.

CALPHAM2 Kinematic hardening shift moment about the 2-direction.

CALPHAM3 Kinematic hardening shift moment about the 3-direction.

Viscous Force Components

CVF1 Viscous force in the 1-direction.

CVF2 Viscous force in the 2-direction.

CVF3 Viscous force in the 3-direction.

CVM1 Viscous moment about the 1-direction.

CVM2 Viscous moment about the 2-direction.

CVM3 Viscous moment about the 3-direction.

Uniaxial Force Components

Connector uniaxial behavior can be defined only in Abaqus/Explicit; therefore, there is no uniaxial force output available in Abaqus/Standard.

CUF1 Uniaxial force in the 1-direction.

CUF2 Uniaxial force in the 2-direction.

CUF3 Uniaxial force in the 3-direction.

CUM1 Uniaxial moment about the 1-direction.

CUM2 Uniaxial moment about the 2-direction.

CUM3 Uniaxial moment about the 3-direction.

Friction Force Components

CSF1 Force due to frictional stress in the 1-direction.

CSF2 Force due to frictional stress in the 2-direction.

CSF3 Force due to frictional stress in the 3-direction.

CSM1 Frictional moment about the 1-direction.

CSM2 Frictional moment about the 2-direction.

CSM3 Frictional moment about the 3-direction.

CSFC Force due to frictional stress in the instantaneous slip direction. Available only for predefined or user-defined coupled friction interactions.

Contact Force Components Generating Friction

CNF1 Contact force generating friction in the 1-direction.

CNF2 Contact force generating friction in the 2-direction.

CNF3 Contact force generating friction in the 3-direction.

CNM1 Contact moment generating friction about the 1-direction.

CNM2 Contact moment generating friction about the 2-direction.

CNM3 Contact moment generating friction about the 3-direction.

CNFC Contact force generating friction in the instantaneous slip direction.

Total Overall Damage Components

CDMG1 Overall damage variable in the 1-direction.

CDMG2 Overall damage variable in the 2-direction.

CDMG3 Overall damage variable in the 3-direction.

CDMGR1 Overall damage variable about the 1-direction.

CDMGR2 Overall damage variable about the 2-direction.

CDMGR3 Overall damage variable about the 3-direction.

Connector Force-Based Damage Initiation Criteria

CDIF1 Connector force-based damage initiation criterion in the 1-direction.

CDIF2 Connector force-based damage initiation criterion in the 2-direction.

CDIF3 Connector force-based damage initiation criterion in the 3-direction.

CDIFR1 Connector force-based damage initiation criterion about the 1-direction.

CDIFR2 Connector force-based damage initiation criterion about the 2-direction.

CDIFR3 Connector force-based damage initiation criterion about the 3-direction.

CDIFC Connector force-based damage initiation criterion in the instantaneous slip direction.

Connector Motion-Based Damage Initiation Criteria

CDIM1 Connector motion-based damage initiation criterion in the 1-direction.

CDIM2 Connector motion-based damage initiation criterion in the 2-direction.

CDIM3 Connector motion-based damage initiation criterion in the 3-direction.

CDIMR1 Connector motion-based damage initiation criterion about the 1-direction.

CDIMR2 Connector motion-based damage initiation criterion about the 2-direction.

CDIMR3 Connector motion-based damage initiation criterion about the 3-direction.

CDIMC Connector motion-based damage initiation criterion in the instantaneous slip direction.

Connector Plastic Motion-Based Damage Initiation Criteria

CDIP1 Connector plastic motion-based damage initiation criterion in the 1-direction.

CDIP2 Connector plastic motion-based damage initiation criterion in the 2-direction.

CDIP3 Connector plastic motion-based damage initiation criterion in the 3-direction.

CDIPR1 Connector plastic motion-based damage initiation criterion about the 1-direction.

CDIPR2 Connector plastic motion-based damage initiation criterion about the 2-direction.

CDIPR3 Connector plastic motion-based damage initiation criterion about the 3-direction.

CDIPC Connector plastic motion-based damage initiation criterion in the instantaneous slip direction.

Connector Lock or Stop Status

CSLST*i* Flags for connector stop and connector lock status $(i = 1, 6)$.

Friction-Related Accumulated Slip

CASU1 Accumulated frictional slip in the 1-direction.

CASU2 Accumulated frictional slip in the 2-direction.

CASU3 Accumulated frictional slip in the 3-direction.

CASUR1 Accumulated frictional rotation about the 1-direction.

CASUR2 Accumulated frictional rotation about the 2-direction.

CASUR3 Accumulated frictional rotation about the 3-direction.

CASUC Accumulated frictional slip in the instantaneous slip direction.

Frictional Instantaneous Velocity in the Slip Direction (Available Only If Friction Is Defined in the Slip Direction)

CIVC Friction-related instantaneous velocity in the slip direction.

Reaction Force Components due to Kinematic Constraints, Connector Locks, Connector Stops, and Prescribed Connector Motion

CRF1 Connector reaction force in the 1-direction.

CRF2 Connector reaction force in the 2-direction.

CRF3 Connector reaction force in the 3-direction.

CRM1 Connector reaction moment about the 1-direction.

CRM2 Connector reaction moment about the 2-direction.

CRM3 Connector reaction moment about the 3-direction.

Connector Concentrated Force Components due to Connector Loads

CCF1 Connector concentrated force in the 1-direction.

CCF2 Connector concentrated force in the 2-direction.

CCF3 Connector concentrated force in the 3-direction.

CCM1 Connector concentrated moment about the 1-direction.

CCM2 Connector concentrated moment about the 2-direction.

CCM3 Connector concentrated moment about the 3-direction.

Relative Position Components

CP1 Relative position in the 1-direction.

CP2 Relative position in the 2-direction.

CP3 Relative position in the 3-direction.

CPR1 Relative angular position in the 1-direction.

CPR2 Relative angular position in the 2-direction.

CPR3 Relative angular position in the 3-direction.

Relative Displacement Components

CU1 Relative displacement in the 1-direction.

CU2 Relative displacement in the 2-direction.

CU3 Relative displacement in the 3-direction.

CUR1 Relative rotation in the 1-direction.

CUR2 Relative rotation in the 2-direction.

CUR3 Relative rotation in the 3-direction.

Constitutive Displacement Components

CCU1 Constitutive displacement in the 1-direction.

CCU2 Constitutive displacement in the 2-direction.

CCU3 Constitutive displacement in the 3-direction.

CCUR1 Constitutive rotation in the 1-direction.

CCUR2 Constitutive rotation in the 2-direction.

CCUR3 Constitutive rotation in the 3-direction.

Relative Velocity Components

CV1 Relative velocity in the 1-direction.

CV2 Relative velocity in the 2-direction.

CV3 Relative velocity in the 3-direction.

CVR1 Relative angular velocity in the 1-direction.

CVR2 Relative angular velocity in the 2-direction.

CVR3 Relative angular velocity in the 3-direction.

Relative Acceleration Components

CA1 Relative acceleration in the 1-direction.

CA2 Relative acceleration in the 2-direction.

CA3 Relative acceleration in the 3-direction.

CAR1 Relative angular acceleration in the 1-direction.

CAR2 Relative angular acceleration in the 2-direction.

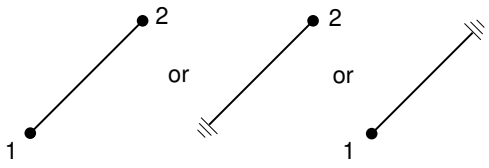
CAR3 Relative angular acceleration in the 3-direction.

Connector Failure Status

CFAILST*i*

Flags for connector failure status $(i = 1, 6)$.

Node Ordering on Elements



Connection Types

Abaqus provides a library of basic, assembled, and complex connections.

The connection-type library contains:

- translational basic connection components, which affect translational degrees of freedom at both nodes and may affect rotational degrees of freedom at the first node or at both nodes on the connector element;
- rotational basic connection components, which affect only rotational degrees of freedom at both nodes on the connector element;
- specialized rotational basic connection components, which in addition to rotational degrees of freedom affect other degrees of freedom at the nodes on the connector element;
- assembled connections, which are predefined combinations of translational and rotational or translational and specialized rotational basic connection components; and
- complex connections, which affect a combination of degrees of freedom at the nodes on the connector element and cannot be combined with any other connection component.

In this section:

- *Basic, Assembled, and Complex Connections*
- *Using the Connection-Type Library*
- *Connection-Type Library*

Basic, Assembled, and Complex Connections

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

The connection-type library contains basic, assembled, and complex connections.

Basic Connections

Basic connection components are divided into three categories:

- Translational basic connection components affect translational degrees of freedom at both nodes and may affect rotational degrees of freedom at the first node or at both nodes;
- Rotational basic connection components affect only rotational degrees of freedom at both nodes; and
- Specialized rotational basic connection components affect rotational degrees of freedom as well as other degrees of freedom at the nodes.

Only one translational basic connection component and one rotational or specialized rotational basic connection component can be used in the definition of a connector element. If a more complicated connection requires more basic connection components than this, use multiple connector elements attached to the same nodes.

Translational Basic Connection Components

The following basic connection components affect translational degrees of freedom at both node *a* and node *b*. Some of these connector components (ACCELEROMETER, CARTESIAN, PROJECTION CARTESIAN, RADIAL-THRUST,SLIDE-PLANE and SLOT) affect rotational degrees of freedom at node *a* or at both node *a* and node *b* (see *Degree of Freedom Activation and Corotation of Connection Directions*). Any basic connection component from this list can be used to define the translational behavior of a connector element.

ACCELEROMETER

Provide a connection between two nodes to measure the relative acceleration, velocity, and position of a body in a local coordinate system. This connection type is available only in Abaqus/Explicit. If it is defined in an Abaqus/Standard model, it will be converted internally to a CARTESIAN connector type.

AXIAL

Provide a connection between two nodes that acts along the line connecting the nodes.

CARTESIAN

Provide a connection between two nodes that allows independent behavior in three local Cartesian directions that follow the system at node *a*.

JOIN

Join the position of two nodes.

LINK

Provide a pinned rigid link between two nodes to keep the distance between the two nodes constant.

PROJECTION CARTESIAN

Provide a connection between two nodes that allows independent behavior in three local Cartesian directions that follow the system at both nodes *a* and *b*.

RADIAL-THRUST

Provide a connection between two nodes that allows different behavior for radial and thrust displacements.

SLIDE-PLANE

Provide a slide-plane connection to make the position of the second node remain on a plane defined by the orientation of the first node and the initial position of the second node.

SLOT

Provide a slot connection to make the position of the second node remain on a line defined by the orientation of the first node and the initial position of the second node.

Rotational Basic Connection Components

The following basic connection components affect only rotational degrees of freedom at the nodes in the connection. Any basic connection component from this list can be used to define the rotational behavior of a connector element.

ALIGN

Provide a connection between two nodes that aligns their local directions.

CARDAN

Provide a rotational connection between two nodes parameterized by Cardan (or Bryant) angles.

CONSTANT VELOCITY

Provide a constant velocity connection between two nodes.

EULER

Provide a rotational connection between two nodes parameterized by Euler angles.

FLEXION-TORSION

Provide a connection between two nodes that allows different behavior for flexural and torsional rotations.

PROJECTION FLEXION-TORSION

Provide a connection between two nodes that allows different behavior for two flexural rotations and one torsional rotation.

REVOLUTE

Provide a revolute connection between two nodes.

ROTATION

Provide a rotational connection between two nodes parameterized by the rotation vector.

ROTATION-ACCELEROMETER

Provide a connection between two nodes to measure the relative angular acceleration, velocity, and position of a body in a local coordinate system. This connection type is available only in Abaqus/Explicit. If it is defined in an Abaqus/Standard model, it will be converted internally to a CARDAN connector type.

UNIVERSAL

Provide a universal connection between two nodes.

Specialized Rotational Basic Connection Components

The following basic connection component affects rotational and other non-translational degrees of freedom at the nodes in the connection. The specialized rotational basic connection component can be combined with translational basic connection components.

FLOW-CONVERTER

Provide a means of converting the material flow (degree of freedom 10) at a connector node into a rotation.

Assembled Connections

Assembled connections, which are predefined combinations of translational and rotational or translational and specialized rotational basic connection components, are included for convenience. The equivalent basic connection components used for each assembled connection are listed in parentheses.

BEAM

Provide a rigid beam connection between two nodes. (JOIN + ALIGN)

BUSHING

Provide a connection between two nodes that allows independent behavior in three local Cartesian directions that follow the system at both nodes *a* and *b* and that allows different behavior in two flexural rotations and one torsional rotation. (PROJECTION CARTESIAN + PROJECTION FLEXION-TORSION)

CVJOINT

Join the position of two nodes, and provide a constant velocity connection between their rotational degrees of freedom. (JOIN + CONSTANT VELOCITY)

CYLINDRICAL

Provide a slot connection between two nodes, and constrain the rotations by a revolute connection. (SLOT + REVOLUTE)

HINGE

Join the position of two nodes, and provide a revolute connection between their rotational degrees of freedom. (JOIN + REVOLUTE)

PLANAR

Provide a slide-plane connection between two nodes with a revolute connection about the normal direction to the plane. The PLANAR connection creates a local two-dimensional system in three-dimensional analyses. (SLIDE-PLANE + REVOLUTE)

RETRACTOR

Join the position of two nodes, and convert material flow into rotation. (JOIN + FLOW-CONVERTER)

TRANSLATOR

Provide a slot connection between two nodes, and align their three local axis directions. (SLOT + ALIGN)

UJOINT

Join the position of two nodes, and provide a universal connection between their rotational degrees of freedom at the nodes. (JOIN + UNIVERSAL)

WELD

Join the position of two nodes, and align their three local axis directions. (JOIN + ALIGN)

Complex Connections

Complex connection components affect a combination of degrees of freedom at the nodes in the connection and cannot be combined with other connection components. They typically model highly coupled physical connections.

SLIPRING

Model material flow and stretching between two points of a belt system (such as an automotive seat belt).

Using the Connection-Type Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Each connection type is described in the connection-type library.
Each library entry includes a figure, which relates the physical behavior to the idealized model and defines the local coordinate directions. Following the figure, each library entry defines kinematic constraints; constraint forces and moments internal to the connection; components of relative motion available for defining the connector behavior, connector motion, or connector loads (called available components); and kinetic forces and moments conjugate to the available components of relative motion. If appropriate, a discussion of the predicted Coulomb-like friction in the connection is included. Finally, the connection type is summarized in a table.

Connection Figures

A schematic drawing of each connection type is included along with the Abaqus idealization of the connection. The idealization indicates in what sense available components of relative motion are measured and how the nodes' positions and orientation directions define the connection. When orientation directions are used to define the connection, the idealization shows these local directions at the appropriate nodes. If available components of relative motion exist in the connection, they are indicated in the figure as free relative motions. *Figure 1* shows the connection figure for the REVOLUTE connection type, which affects only rotations. It has one available component (the rotation about the shared axis), requires an orientation at node *a*, and allows an optional orientation at node *b*.

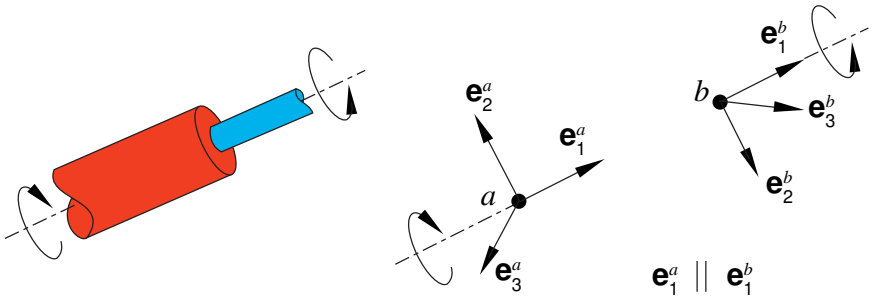


Figure 1: Example connection type: REVOLUTE.

Orientation Directions

The orientation directions at node *a* (the first node on the connector element) are indicated as unit base vectors $\mathbf{e}_i^a$, where $i = \{1, 2, 3\}$. Similarly, the orientation directions at node *b* are indicated as $\mathbf{e}_i^b$. When orientation directions are required at a node, you must define them as described in *Orientations*. If orientation directions are optional but not provided at node *a*, the global directions are used by default. If orientation directions are optional but not provided at node *b*, the orientation directions from node *a* are used by default.

Connector elements activate rotational degrees of freedom at the nodes to which they are attached if they do not exist already and an orientation is permitted at that node. The only exception is connection type JOIN, where an orientation is optional at node *a* but rotation degrees of freedom are not activated.

The orientation directions corotate with the rotation of the node to which they are attached (with the exception of connection type JOIN, which uses fixed directions when rotation degrees of freedom are not active at node *a*). If there are no elements with rotational degrees of freedom attached to the node, rotational multi-point constraints, or rotational equations, you must ensure that sufficient rotational boundary conditions are provided to avoid numerical singularities associated with unconstrained rotational degrees of freedom.

Components of Relative Motion and Connector Forces and Moments

The six components of relative motion, denoted u_i and ur_i for $i = \{1, 2, 3\}$, are defined in the description for each connection, where needed. These components include constrained and available components of relative motion. Forces and moments are denoted $\mathbf{f}$ and $\mathbf{m}$. These quantities are either constraint forces and moments, which enforce the kinematic constraints on the constrained components of relative motion, or kinetic forces and moments, which are the work conjugate variables to the available components of relative motion. For example, the REVOLUTE connection type has one available component of relative motion, ur_1, and two kinematic rotation constraints (equivalent to setting two rotation components, ur_2 and ur_3, to zero). Conjugate to the available rotation component is the kinetic moment m_1 acting about the local $\mathbf{e}_1^a$-direction.

In general, kinetic forces and moments include the effects of connector behaviors, such as elastic springs, viscous damping, friction, and reaction forces and moments due to connector stops and locks. For constitutive response defined as a function of displacement or rotation, the initial position may not correspond with the reference position where constitutive forces and moments are zero. You can define reference lengths and angles (given in degrees) for connector behavior as described in *Defining Reference Lengths and Angles for Constitutive Response*.

These reference quantities define u_i^{mat} and ur_i^{mat}, the connector constitutive displacements and rotations. These constitutive displacements and rotations are used only to define constitutive response and differ from the relative displacements and rotations measured in the connector elements only when you define the reference lengths or angles.

As an example, if the REVOLUTE connection included linear spring and dashpot behavior combined with a connector stop,

$$m_1 = K_1 ur_1^{mat} + C_1 ur_1 + RM_1,$$

where K_1 is the spring stiffness, C_1 is the dashpot coefficient, and RM_1 is the reaction moment caused by the connector stop. In the REVOLUTE connection there are two constraint moment components, m_2 about $\mathbf{e}_2^a$ and m_3 about $\mathbf{e}_3^a$.

Interpreting Connector Forces and Moments

The kinematic constraint and kinetic forces and moments are always computed as work conjugates of the kinematics in the connector (components of relative motion). In most connection types one direct consequence is that the constraint forces (and moments) in connectors are reported as the forces (and moments) applied at the second node but in the local system associated with the first node. Since the kinematics are complex in many of the connection types, the connector forces and moments can be somewhat surprising on first inspection. For example, consider the case of a HINGE connection defined with the local $\mathbf{e}_1^a$-direction aligned with the global *X*-direction and the local $\mathbf{e}_2^a$-direction aligned with the global Y-direction. Assume that the second connector node is grounded and that the first node is subjected to a concentrated load along the global *Y*-direction. If the only available relative rotation in the HINGE is constrained with a zero-valued connector motion, the second node does not rotate with respect to the first node and the connector reaction force along the local $\mathbf{e}_2^a$-direction

matches the applied load while the other two connector reaction forces are zero. However, if a nonzero connector motion is specified, the first connector reaction force is still zero while both the second and third connector reaction forces are nonzero and only the vector-norm of these two forces matches the applied load. In both cases the only nonzero nodal reaction force at the second connector node is the one in the global *Y*-direction, as dictated by the equilibrium in a free body diagram. Hence, the connector reaction forces and nodal reaction forces are not equivalent in most cases.

Coulomb-Like Friction Behavior

Coulomb-like friction behavior is possible for any connection type that has available components of relative motion; see *Connector Friction Behavior* for details. Friction behavior requires a "tangent" direction (the direction in which slipping may occur) and a "normal" direction (the direction perpendicular to the contacting surfaces). In the most general case you define the normal force causing friction in the connector. However, Abaqus predefines friction behavior for a limited number of connection types, as discussed in the connection-type library in this section. In these predefined friction cases you do not have to define the contact normal force.

Summary Table

Each connection library entry includes a table summarizing the connection type. This summary table indicates whether the connection type is basic, assembled, or complex. It gives the kinematic constraints; constraint force or moment components; available components of relative motion; "kinetic" force or moment components following from the constitutive behavior in the available components of relative motion; which orientation directions are required, optional, or ignored; how connector stops limit the available components of relative motion; what reference lengths and angles are used to define the constitutive behavior; what parameters are used for predefined Coulomb-like friction; and how the contact normal forces are defined by Abaqus in association with predefined Coulomb-like friction.

Connection-Type Library

This section provides a reference to the connection types available in Abaqus.

In this section:

- *ACCELEROMETER*
- *ALIGN*
- *AXIAL*
- *BEAM*
- *BUSHING*
- *CARDAN*
- *CARTESIAN*
- *CONSTANT VELOCITY*
- *CVJOINT*
- *CYLINDRICAL*
- *EULER*
- *FLEXION-TORSION*
- *FLOW-CONVERTER*
- *HINGE*
- *JOIN*
- *LINK*
- *PLANAR*
- *PROJECTION CARTESIAN*
- *PROJECTION FLEXION-TORSION*
- *RADIAL-THRUST*
- *RETRACTOR*
- *REVOLUTE*
- *ROTATION*
- *ROTATION-ACCELEROMETER*
- *SLIDE-PLANE*
- *SLIPRING*
- *SLOT*
- *TRANSLATOR*
- *UJOINT*
- *UNIVERSAL*
- *WELD*

ACCELEROMETER

Products: Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type ACCELEROMETER provides a convenient way to measure the relative position, velocity, and acceleration of a body in a local coordinate system.
All kinematic quantities are measured relative to node *a*. The position, displacement, velocity, and acceleration are reported in the coordinate system of node *b*. Each node of the connector can translate and rotate independently, although fixing the first of the two nodes to ground is more common. With the first node fixed, connection type ACCELEROMETER provides a convenient way to measure the local components of the velocity and acceleration in a coordinate system fixed to a moving body (for example, an accelerometer).

Connection type ACCELEROMETER is available only in Abaqus/Explicit. It is the translation counterpart to connection type ROTATION-ACCELEROMETER, which measures relative angular position, velocity, and acceleration. ACCELEROMETER connections cannot be used in two-dimensional and axisymmetric analyses in Abaqus/Explicit.

Description

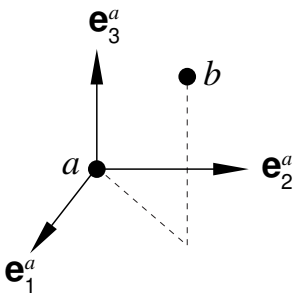


Figure 1: Connection type ACCELEROMETER.

The ACCELEROMETER connection does not impose kinematic constraints. It defines three local directions at node *a* and three local directions at node *b*. The ACCELEROMETER connection's formulation is similar to that for the CARTESIAN connection. The ACCELEROMETER connection measures the position of node *b* relative to node *a*

$$x = \mathbf{e}_1^a \cdot (\mathbf{x}_b - \mathbf{x}_a) \ ; \quad y = \mathbf{e}_2^a \cdot (\mathbf{x}_b - \mathbf{x}_a) \ ; \quad \text{and} \quad z = \mathbf{e}_3^a \cdot (\mathbf{x}_b - \mathbf{x}_a) \ .$$

The coordinate system at node *b* rotates with the rotation of node *b*, and the kinematic quantities are reported in this rotated coordinate system as shown in *Figure 2*.

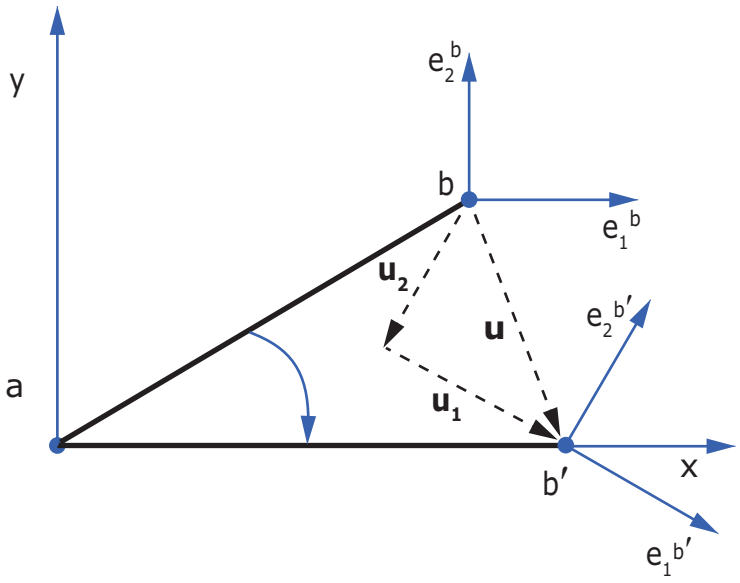


Figure 2: Kinematic quantities in rotated coordinate system.

There are no available components of relative motion for the ACCELEROMETER connection. The connector displacement components are

$$u_1 = x - x_0 \ , \quad u_2 = y - y_0 \ , \quad \text{and} \quad u_3 = z - z_0 \ ,$$

where x_0, y_0, and z_0 are the initial coordinates of node b relative to node a.

The ACCELEROMETER connection measures velocity and acceleration in the local directions at node a as if node a were an inertial frame. In contrast to the CARTESIAN connection, the ACCELEROMETER connection reports all the computed quantities (such as displacement, position, velocity, and acceleration) in the local directions at node b.

Let T_{ij} be the transformation from $\mathbf{e}_i^a$ to $\mathbf{e}_i^b$. Then the ACCELEROMETER connection measures velocity and acceleration as

$$v_i = T_{ij} \frac{\mathrm{d}u_j}{\mathrm{d}t} \quad \text{and} \quad a_i = T_{ij} \frac{\mathrm{d}^2 u_j}{\mathrm{d}t^2} \ ,$$

where the derivatives above are time derivatives in a system moving with $\mathbf{e}_i^a$.

In two-dimensional and axisymmetric analyses $u_3 = 0$.

Summary

ACCELEROMETER	
Basic, assembled, or complex:	Basic
Kinematic constraints:	None
Constraint force output:	None
Available components:	None
Kinetic force output:	None
Orientation at *a*:	Optional
Orientation at *b*:	Optional

ACCELEROMETER	
Connector stops:	None
Constitutive reference lengths:	None
Predefined friction parameters:	None
Contact force for predefined friction:	None

ALIGN

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type ALIGN provides a connection between two nodes where all three local directions are aligned. If both local axes are given and do not align initially, their initial relative angular position is held constant.

Description

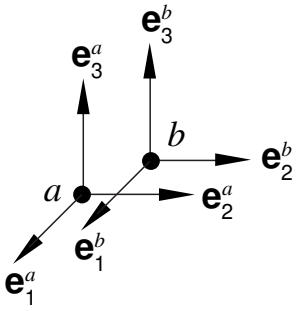


Figure 1: Connection type ALIGN.

The ALIGN connection imposes kinematic constraints only. The local directions at node b are set equal to those at node a. If the local directions do not align initially, the ALIGN connection holds fixed the Cardan angles between the local orientation directions at node b, $\{\mathbf{e}_1^b, \mathbf{e}_2^b, \mathbf{e}_3^b\}$, and those at node a, $\{\mathbf{e}_1^a, \mathbf{e}_2^a, \mathbf{e}_3^a\}$. These fixed angular positions are the connector position output quantities. See connection type *CARDAN* for a definition of Cardan angles and for the constraint moment if the local directions do not align.

The constraint moment enforcing the alignment of the local directions is

$$\overline{\mathbf{m}} = m_1 \mathbf{e}_1^a + m_2 \mathbf{e}_2^a + m_3 \mathbf{e}_3^a \, .$$

In two-dimensional analysis $m_1 = m_2 = 0$.

Summary

ALIGN	
Basic, assembled, or complex:	Basic
Kinematic constraints:	$ur_1 = 0, ur_2 = 0, ur_3 = 0$
Constraint moment output:	m_1, m_2, m_3

ALIGN	
Available components:	None
Kinetic moment output:	None
Orientation at *a*:	Optional
Orientation at *b*:	Optional
Connector stops:	None
Constitutive reference angles:	None
Predefined friction parameters:	None
Contact force for predefined friction:	None

AXIAL

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type AXIAL provides a connection between two nodes where the relative displacement is along the line separating the two nodes.
The AXIAL connection models discrete physical connections such as axial springs, axial dashpots, or node-to-node (gap-like) contact.

Description

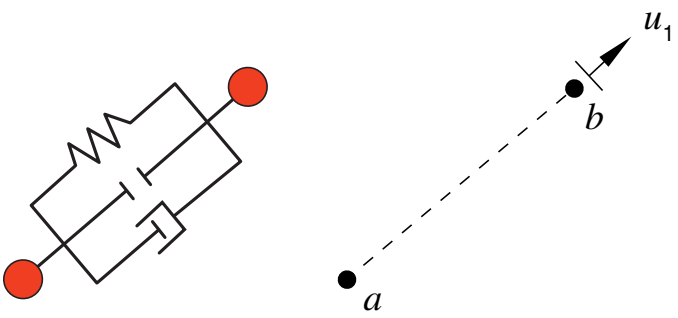


Figure 1: Connection type AXIAL.

The AXIAL connection does not constrain any component of relative motion. The distance between nodes *a* and *b* is

$$l = \|\mathbf{x}_b - \mathbf{x}_a\| .$$

The available component of relative motion, u_1, acts along the line connecting the two nodes, measures the change in distance separating the two nodes, and is defined as

$$u_1 = l - l_0 ,$$

where l_0 is the initial distance from node *a* to *b*. The connector constitutive displacement is

$$u_1^{mat} = l - l_1^{ref} .$$

The kinetic force is

$$\mathbf{f}_{axial} = f_1 \mathbf{q} , \quad \text{where} \quad \mathbf{q} = \frac{1}{\|\mathbf{x}_b - \mathbf{x}_a\|} (\mathbf{x}_b - \mathbf{x}_a) .$$

In Abaqus/Standard an optional orientation can be provided at one of the nodes in an AXIAL connection to provide direction for the force if the nodes are coincident or when one of the nodes is a “ground node.” If the orientation is provided at both of the coincident nodes, the orientation at the first node in the connectivity will be used. The orientation

definitions remain fixed during the analysis and will be ignored when the two nodes separate. Rotational degrees of freedom are not activated for connection type AXIAL.

Summary

AXIAL	
Basic, assembled, or complex:	Basic
Kinematic constraints:	None
Constraint force output:	None
Available components:	u_1
Kinetic force output:	f_1
Orientation at *a*:	Optional
Orientation at *b*:	Optional
Connector stops:	$l_1^{min} \le l \le l_1^{max}$
Constitutive reference lengths:	l_1^{ref}
Predefined friction parameters:	None
Contact force for predefined friction:	None

BEAM

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type BEAM provides a rigid beam connection between two nodes.

Description

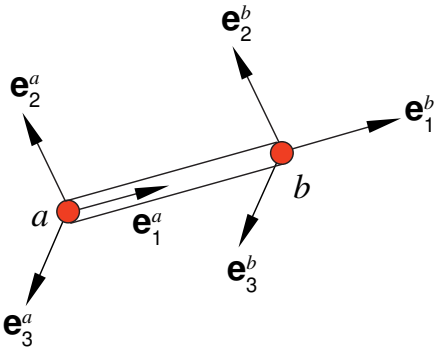


Figure 1: Connection type BEAM.

Connection type BEAM imposes kinematic constraints and uses local orientation definitions equivalent to combining connection types JOIN and ALIGN.

Summary

BEAM	
Basic, assembled, or complex:	Assembled
Kinematic constraints:	JOIN + ALIGN
Constraint force and moment output:	$f_1, f_2, f_3, m_1, m_2, m_3$
Available components:	None
Kinetic force and moment output:	None
Orientation at *a*:	Optional
Orientation at *b*:	Optional
Connector stops:	None
Constitutive reference lengths and angles:	None

BEAM	
Predefined friction parameters:	None
Contact force for predefined friction:	None

BUSHING

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type BUSHING provides a bushing-like connection between two nodes.
It cannot be used in two-dimensional or axisymmetric analyses.

Description

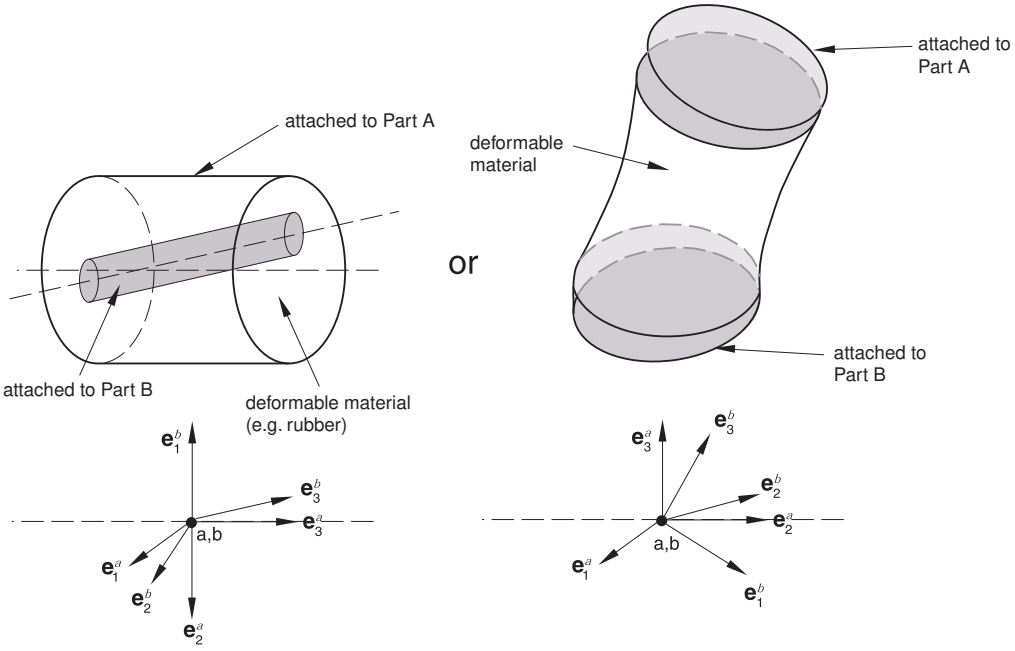


Figure 1: Connection type BUSHING.

Connection type BUSHING does not constrain any components of relative motion and uses local orientation definitions equivalent to combining connection types PROJECTION CARTESIAN and PROJECTION FLEXION-TORSION.

Limitation

In Abaqus/Explicit, if the total flexion angle vanishes, connector motion cannot be applied to the flexion component of motion.

Summary

BUSHING	
Basic, assembled, or complex:	Assembled
Kinematic constraints:	None
Constraint force and moment output:	None
Available components:	$u_1, u_2, u_3, ur_1, ur_2, ur_3$
Kinetic force and moment output:	$f_1, f_2, f_3, m_1, m_2, m_3$
Orientation at *a*:	Required
Orientation at *b*:	Optional
Connector stops:	$l_1^{min} \le x \le l_1^{max}$,
	$l_2^{min} \le y \le l_2^{max}$,
	$l_3^{min} \le z \le l_3^{max}$
	$\theta_1^{min} \le \alpha_1 \le \theta_1^{max}$,
	$\theta_2^{min} \le \alpha_2 \le \theta_2^{max}$,
	$\theta_3^{min} \le \beta \le \theta_3^{max}$
Constitutive reference lengths and angles:	$l_1^{ref}, l_2^{ref}, l_3^{ref}$
	$\theta_1^{ref}, \theta_2^{ref}, \theta_3^{ref}$
Predefined friction parameters:	None
Contact force for predefined friction:	None

CARDAN

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type CARDAN provides a rotational connection between two nodes where the relative rotation between the nodes is parameterized by Cardan (or Bryant) angles.
A Cardan-angle parameterization of finite rotations is also called a 1–2–3 or yaw-pitch-roll parameterization. Connection type CARDAN cannot be used in two-dimensional or axisymmetric analysis.

When connection type CARDAN is used with connector behavior, the relative rotation axis with the highest resistance to rotational motion should be assigned to the second component of relative rotation (component number 5) to avoid "gimbal lock," a singularity in the rotation parameterization for relative rotation angles $\beta = \pm\pi/2$.

Description

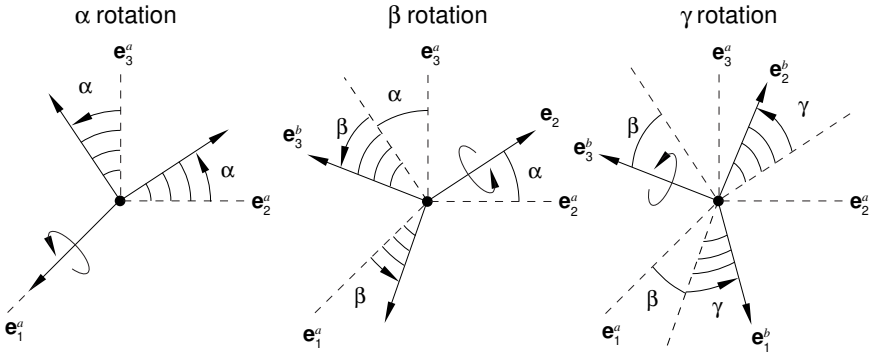


Figure 1: Connection type CARDAN.

The CARDAN connection does not impose kinematic constraints. A CARDAN connection is a finite rotation connection where the local directions at node *b* are parameterized in terms of Cardan (or Bryant) angles relative to the local directions at node *a*. Local directions $\{\mathbf{e}_1^b, \mathbf{e}_2^b, \mathbf{e}_3^b\}$ are positioned relative to $\{\mathbf{e}_1^a, \mathbf{e}_2^a, \mathbf{e}_3^a\}$ by three successive finite rotations α, β, and γ as follows:

1. Rotate by α radians about axis $\mathbf{e}_1^a$;
2. Rotate by β radians about the intermediate 2-axis, $\mathbf{e}_2 = \cos\alpha\mathbf{e}_2^a + \sin\alpha\mathbf{e}_3^a$; and
3. Rotate by γ radians about axis $\mathbf{e}_3^b$.

Rotation angle β should be moderate (magnitude less than $\pi/2$), whereas α and γ may be arbitrarily large (i.e., magnitude greater than 2π). The Cardan angles are determined by the local directions as

$$\alpha = -\tan^{-1}\left(\frac{\mathbf{e}_2^a \cdot \mathbf{e}_3^b}{\mathbf{e}_3^a \cdot \mathbf{e}_3^b}\right) + m\pi;$$

$$\beta = \sin^{-1}\left(\mathbf{e}_1^a \cdot \mathbf{e}_3^b\right), \quad -\frac{\pi}{2} < \beta < \frac{\pi}{2};$$

$$\gamma = -\tan^{-1}\left(\frac{\mathbf{e}_1^a \cdot \mathbf{e}_2^b}{\mathbf{e}_1^a \cdot \mathbf{e}_1^b}\right) + n\pi.$$

Here, *m* and *n* are integers that account for rotations with a magnitude greater than π.

The three available components of relative motion in the CARDAN connection are the changes in the Cardan angles positioning the local directions at node *b* relative to the local directions at node *a*. Therefore,

$$ur_1 = \alpha - \alpha_0; \quad ur_2 = \beta - \beta_0; \quad \text{and} \quad ur_3 = \gamma - \gamma_0;$$

where α_0, β_0, and γ_0 are the initial Cardan angles. The connector constitutive rotations are

$$ur_1^{mat} = \alpha - \theta_1^{ref}; \quad ur_2^{mat} = \beta - \theta_2^{ref}; \quad \text{and} \quad ur_3^{mat} = \gamma - \theta_3^{ref}.$$

The kinetic moment in a CARDAN connection is determined from the three component relationships:

$$\mathbf{m}_{Cardan} = m_1 \mathbf{e}_1^a + m_2\left(\cos\alpha \mathbf{e}_2^a + \sin\alpha \mathbf{e}_3^a\right) + m_3 \mathbf{e}_3^b.$$

Summary

CARDAN	
Basic, assembled, or complex:	Basic
Kinematic constraints:	None
Constraint moment output:	None
Available components:	ur_1, ur_2, ur_3
Kinetic moment output:	m_1, m_2, m_3
Orientation at *a*:	Required
Orientation at *b*:	Optional
Connector stops:	$\theta_1^{min} \le \alpha \le \theta_1^{max}$,
	$\theta_2^{min} \le \beta \le \theta_2^{max}$,
	$\theta_3^{min} \le \gamma \le \theta_3^{max}$
Constitutive reference angles:	$\theta_1^{ref}, \theta_2^{ref}, \theta_3^{ref}$
Predefined friction parameters:	None
Contact force for predefined friction:	None

CARTESIAN

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type CARTESIAN provides a connection between two nodes where the change in position is measured in three local connection directions for node *a*.

Description

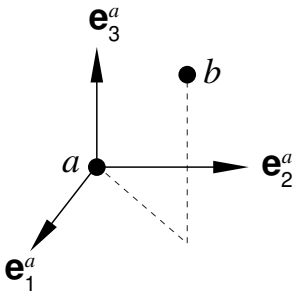


Figure 1: Connection type CARTESIAN.

The CARTESIAN connection does not impose kinematic constraints. It defines three local directions $\{\mathbf{e}_1^a, \mathbf{e}_2^a, \mathbf{e}_3^a\}$ at node *a* and measures the change in position of node *b* along these local coordinate directions. The local directions at node *a* follow the rotation of node *a*.

The position of node *b* relative to node *a* is

$$x = \mathbf{e}_1^a \cdot (\mathbf{x}_b - \mathbf{x}_a) \ ; \quad y = \mathbf{e}_2^a \cdot (\mathbf{x}_b - \mathbf{x}_a) \ ; \quad \text{and} \quad z = \mathbf{e}_3^a \cdot (\mathbf{x}_b - \mathbf{x}_a) \ .$$

The available components of relative motion are

$$u_1 = x - x_0 \ ; \quad u_2 = y - y_0 \ ; \quad \text{and} \quad u_3 = z - z_0 \ ;$$

where x_0, y_0, and z_0 are the initial coordinates of node *b* relative to the local coordinate system at node *a*. The connector constitutive displacements are

$$u_1^{mat} = x - l_1^{ref} \ ; \quad u_2^{mat} = y - l_2^{ref} \ ; \quad \text{and} \quad u_3^{mat} = z - l_3^{ref} \ .$$

The kinetic force is

$$\mathbf{f}_{Cart} = f_1 \mathbf{e}_1^a + f_2 \mathbf{e}_2^a + f_3 \mathbf{e}_3^a \ .$$

In two-dimensional analysis $z = 0$, $u_3 = 0$, $u_3^{mat} = 0$, and $f_3 = 0$.

Summary

CARTESIAN	
Basic, assembled, or complex:	Basic
Kinematic constraints:	None
Constraint force output:	None
Available components:	u_1, u_2, u_3
Kinetic force output:	f_1, f_2, f_3
Orientation at *a*:	Optional
Orientation at *b*:	Ignored
Connector stops:	$l_1^{min} \le x \le l_1^{max}$,
	$l_2^{min} \le y \le l_2^{max}$,
	$l_3^{min} \le z \le l_3^{max}$
Constitutive reference lengths:	$l_1^{ref}, l_2^{ref}, l_3^{ref}$
Predefined friction parameters:	None
Contact force for predefined friction:	None

CONSTANT VELOCITY

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type CONSTANT VELOCITY provides the rotational part of connection type CVJOINT. It cannot be used in two-dimensional or axisymmetric analysis. Furthermore, the connection type does not have available components of relative motion. To include connector behavior in flexural motion, use connection type FLEXION-TORSION with the torsion angle set to zero.

This connection type models physical connectors that under certain conditions transmit a constant spinning velocity about misaligned shafts.

Description



Figure 1: Connection type CONSTANT VELOCITY.

The shaft direction at node a is $\mathbf{e}_3^a$, and the shaft direction at node b is $\mathbf{e}_3^b$. The constant velocity constraint is stated as follows. In any configuration there are two unit length orthogonal vectors $\mathbf{b}_1$ and $\mathbf{b}_2$ in the plane perpendicular to the shaft at node b. These vectors can be written

$$\mathbf{b}_1 = \cos\beta\mathbf{e}_1^b + \sin\beta\mathbf{e}_2^b \quad \text{and} \quad \mathbf{b}_2 = -\sin\beta\mathbf{e}_1^b + \cos\beta\mathbf{e}_2^b\,.$$

The angle β is chosen such that

$$\mathbf{e}_1^a \cdot \mathbf{b}_2 = \mathbf{e}_2^a \cdot \mathbf{b}_1\,.$$

The constant velocity constraint requires that the angle β is constant at all times. The constant velocity constraint is equivalent to constraining the torsion angle to be constant in a FLEXION-TORSION connection.

The name “constant velocity” for this connection type derives from the following property. If the angular velocities of the two shafts, $\mathbf{w}_a$ and $\mathbf{w}_b$, have components only along each shaft, respectively, and in the direction of the normal to the plane containing the two shafts (that is, along the $\mathbf{e}_3^b \times \mathbf{e}_3^a$ direction), the components of angular velocity along the respective shaft directions are equal:

$$\mathbf{w}_a \cdot \mathbf{e}_3^a = \mathbf{w}_b \cdot \mathbf{e}_3^b\,.$$

Hence, the “spinning” angular velocity component is the same about each shaft.

The constraint moment imposing the constant velocity constraint has a single component about the average shaft direction $\mathbf{e}_3^a + \mathbf{e}_3^b$ and is written

$$\overline{\mathbf{m}} = m_2 \frac{(\mathbf{e}_3^a + \mathbf{e}_3^b)}{\|\mathbf{e}_3^a + \mathbf{e}_3^b\|}.$$

Summary

CONSTANT VELOCITY	
Basic, assembled, or complex:	Basic
Kinematic constraints:	$\mathbf{e}_1^a \cdot \mathbf{b}_2 = \mathbf{e}_2^a \cdot \mathbf{b}_1$
Constraint moment output:	m_2
Available components:	None
Kinetic moment output:	None
Orientation at *a*:	Required
Orientation at *b*:	Optional
Connector stops:	None
Constitutive reference angles:	None
Predefined friction parameters:	None
Contact force for predefined friction:	None

CVJOINT

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type CVJOINT joins the position of two nodes and provides a constant velocity constraint between their rotational degrees of freedom.
Connection type CVJOINT cannot be used in two-dimensional or axisymmetric analysis.

Description

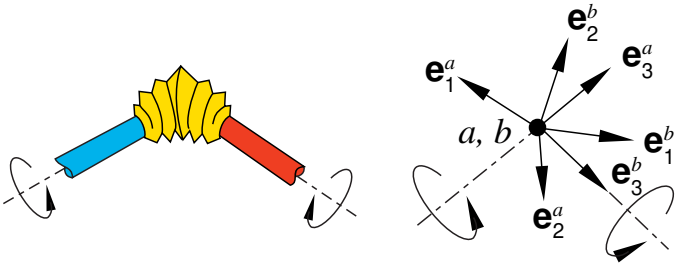


Figure 1: Connection type CVJOINT.

Connection type CVJOINT imposes kinematic constraints and uses local orientation definitions equivalent to combining connection types JOIN and CONSTANT VELOCITY.

Summary

CVJOINT	
Basic, assembled, or complex:	Assembled
Kinematic constraints:	JOIN + CONSTANT VELOCITY
Constraint force and moment output:	f_1, f_2, f_3, m_2
Available components:	None
Kinetic force and moment output:	None
Orientation at *a*:	Required
Orientation at *b*:	Optional
Connector stops:	None
Constitutive reference lengths and angles:	None

CVJOINT	
Predefined friction parameters:	None
Contact force for predefined friction:	None

CYLINDRICAL

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type CYLINDRICAL provides a slot connection between two nodes and a revolute constraint where the free rotation is about the line of the slot.
It cannot be used in two-dimensional or axisymmetric analysis.

Description

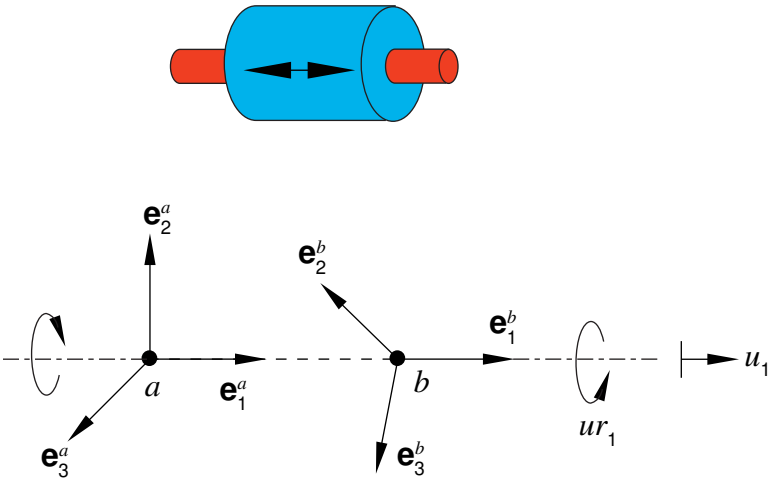


Figure 1: Connection type CYLINDRICAL.

Connection type CYLINDRICAL imposes kinematic constraints and uses local orientation definitions equivalent to combining connection types SLOT and REVOLUTE.

The connector constraint forces and moments reported as connector output depend strongly on the order and the location of the nodes in the connector (see *Connector Behavior*). Since the kinematic constraints are enforced at node *b* (the second node of the connector element), the reported forces and moments are the constraint forces and moments applied at node *b* to enforce the CYLINDRICAL constraint. Thus, in most cases the connector output associated with a CYLINDRICAL connection is best interpreted when node *b* is located at the center of the device enforcing the constraint. This choice is essential when moment-based friction is modeled in the connector since the contact forces are derived on the connector forces and moments, as illustrated below. Proper enforcement of the kinematic constraints is independent of the order or location of the nodes.

Friction

Predefined Coulomb-like friction in the CYLINDRICAL connection defines the friction force (CSFC) along the instantaneous slip direction on the two contacting cylindrical surfaces (the pin and the sleeve) illustrated above. The

table below summarizes the parameters that are used to specify predefined friction in this connection type as discussed in detail next.

The frictional effect is formally written as

$$\Phi = \mathrm{P}(\mathbf{f}) - \mu \mathrm{F_N} \leq 0,$$

where the potential $\mathrm{P}(\mathbf{f})$ represents the magnitude of the frictional tangential tractions in the connector in a direction tangent to the cylindrical surface on which contact occurs, $\mathrm{F_N}$ is the friction-producing normal force on the same cylindrical surface, and μ is the friction coefficient. Frictional stick occurs if $\Phi < 0$; and sliding occurs if $\Phi = 0$, in which case the friction force is $\mu \mathrm{F_N}$.

The normal force $\mathrm{F_N}$ is the sum of a magnitude measure of friction-producing connector forces, $\mathrm{F_C} = g(\mathbf{f})$, and a self-equilibrated internal contact force (such as from a press-fit assembly), $\mathrm{F_C^{int}}$:

$$\mathrm{F_N} = \left| \mathrm{F_C} + \mathrm{F_C^{int}} \right| = \left| g(\mathbf{f}) + \mathrm{F_C^{int}} \right|.$$

The magnitude measure of friction-producing connector contact force, $\mathrm{F_C}$, is defined by summing the following two contributions:

- a radial force contribution, F_r (the magnitude of the constraint forces enforcing the SLOT constraint):

$$F_r = \sqrt{{f_2}^2 + {f_3}^2}, \text{and}$$

- a force contribution from "bending," F_{bend}, obtained by scaling the bending moment, M_{bend} (the magnitude of the constraint moments enforcing the REVOLUTE constraint), by a length factor, as follows:

$$M_{bend} = \sqrt{{m_2}^2 + {m_3}^2},$$

$$F_{bend} = 2\frac{M_{bend}}{L},$$

  where L represents a characteristic overlapping length between the shaft and the outer sleeve in the 1-direction. If L is 0.0, M_{bend} is ignored.

Thus,

$$\mathrm{F_C} = g(\mathbf{f}) = F_r + F_{bend} = \sqrt{{f_2}^2 + {f_3}^2} + \sqrt{(\beta m_2)^2 + (\beta m_3)^2},$$

where $\beta = \frac{2}{L}$.

The magnitude of the frictional tangential moment, $\mathrm{P}(\mathbf{f})$, is computed using

$$\mathrm{P}(\mathbf{f}) = \sqrt{f_1^2 + \left(\frac{m_1}{R}\right)^2},$$

where R is an effective radius of the shaft cross-section in the local 2–3 plane. The potential P represents the magnitude of connector tangential tractions on the cylindrical contact surface due to simultaneous translation and rotation. The instantaneous slip direction is a result of combined motion in these directions.

Summary

CYLINDRICAL	
Basic, assembled, or complex:	Assembled
Kinematic constraints:	SLOT + REVOLUTE
Constraint force and moment output:	f_2, f_3, m_2, m_3
Available components:	u_1, ur_1
Kinetic force and moment output:	f_1, m_1
Orientation at *a*:	Required
Orientation at *b*:	Optional
Connector stops:	$l_1^{min} \le l \le l_1^{max}$
	$\theta_1^{min} \le \alpha \le \theta_1^{max}$
Constitutive reference lengths and angles:	l_1^{ref}, θ_1^{ref}
Predefined friction parameters:	Required: *R*; optional: *L*, $\mathbf{F}_{\mathrm{C}}^{\mathrm{int}}$
Contact force for predefined friction:	$\mathbf{F}_{\mathrm{C}}$

EULER

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type EULER provides a rotational connection between two nodes where the total relative rotation between the nodes is parameterized by Euler angles.
An Euler-angle parameterization of finite rotations is also called a 3–1–3 or precession-nutation-spin parameterization.
Connection type EULER cannot be used in two-dimensional or axisymmetric analysis.

Description

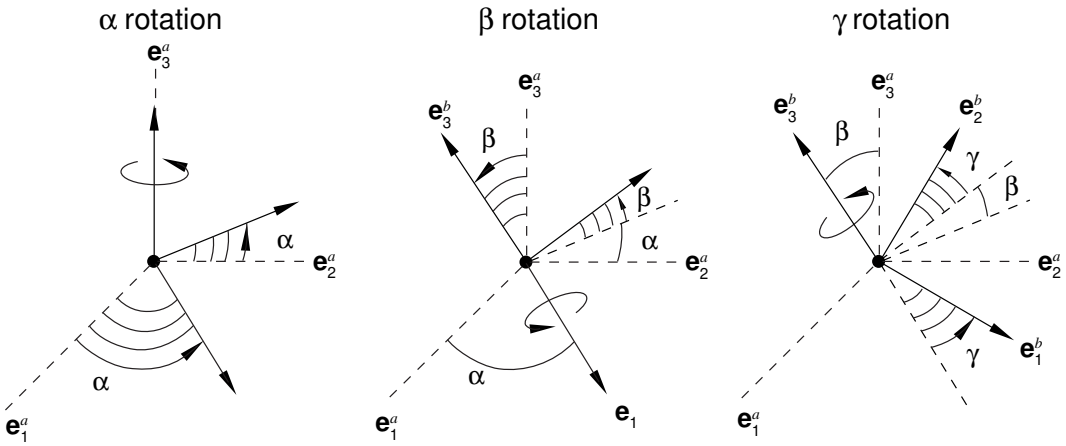


Figure 1: Connection type EULER.

The EULER connection does not impose kinematic constraints. An EULER connection is a finite rotation connection where the local directions at node *b* are parameterized in terms of Euler angles relative to the local directions at node *a*. Local directions $\{\mathbf{e}_1^b, \mathbf{e}_2^b, \mathbf{e}_3^b\}$ are positioned relative to $\{\mathbf{e}_1^a, \mathbf{e}_2^a, \mathbf{e}_3^a\}$ by three successive finite rotations α, β, and γ as follows:

1. Rotate by α radians about axis $\mathbf{e}_3^a$;
2. Rotate by β radians about the intermediate 1-axis, $\mathbf{e}_1 = \cos\alpha\mathbf{e}_1^a + \sin\alpha\mathbf{e}_2^a$;
3. Rotate by γ radians about axis $\mathbf{e}_3^b$.

The Euler angles are determined by the local directions as

$$\alpha = -\tan^{-1}\left(\frac{\mathbf{e}_1^a \cdot \mathbf{e}_3^b}{\mathbf{e}_2^a \cdot \mathbf{e}_3^b}\right) + i\pi;$$

$$\beta = \cos^{-1}\left(\mathbf{e}_3^a \cdot \mathbf{e}_3^b\right) + j\pi;$$
$$\gamma = \tan^{-1}\left(\frac{\mathbf{e}_3^a \cdot \mathbf{e}_1^b}{\mathbf{e}_3^a \cdot \mathbf{e}_2^b}\right) + k\pi.$$

Here i, j, and k are integers that account for rotations with magnitudes greater than π. Initially, the intermediate rotation angle β is chosen in the interval $0 \le \beta \le \pi$.

If the intermediate rotation is an even multiple of π, $\beta = 2m\pi$, where $m = 0, \pm 1, \pm 2, \ldots$, the other two Euler angles become non-unique. In this case

$$\alpha + \gamma = \tan^{-1}\left(\frac{\mathbf{e}_2^a \cdot \mathbf{e}_1^b}{\mathbf{e}_1^a \cdot \mathbf{e}_1^b}\right) + n\pi.$$

Similarly, if the intermediate rotation is an odd multiple of π, $\beta = (2m+1)\pi$, where $m = 0, \pm 1, \pm 2, \ldots$, the other two Euler angles become nonunique as well. In this case

$$\alpha - \gamma = \tan^{-1}\left(\frac{\mathbf{e}_2^a \cdot \mathbf{e}_1^b}{\mathbf{e}_1^a \cdot \mathbf{e}_1^b}\right) + n\pi.$$

In both of these cases a singularity results in the rotation parameterization when the $\mathbf{e}_3^a$ and $\mathbf{e}_3^b$ axes align. The EULER connection should be used in such a way that these axes do not align throughout the computation. For a singularity-free condition Abaqus will choose α and γ such that a smooth parameterization results for the above values of the intermediate angle β.

The available components of relative motion in the EULER connection are the changes in the Euler angles that position the local directions at node b relative to the local directions at node a. Therefore,

$$ur_1 = \alpha - \alpha_0\ ; \quad ur_2 = \beta - \beta_0\ ; \quad \text{and} \quad ur_3 = \gamma - \gamma_0;$$

where α_0, β_0, and γ_0 are the initial Euler angles. The connector constitutive rotations are

$$ur_1^{mat} = \alpha - \theta_1^{ref}\ ; \quad ur_2^{mat} = \beta - \theta_2^{ref}\ ; \quad \text{and} \quad ur_3^{mat} = \gamma - \theta_3^{ref}.$$

The kinetic moment in a EULER connection is determined from the three component relationships:

$$\mathbf{m}_{Euler} = m_1\mathbf{e}_3^a + m_2\left(\cos\alpha\mathbf{e}_1^a + \sin\alpha\mathbf{e}_2^a\right) + m_3\mathbf{e}_3^b.$$

Summary

EULER	
Basic, assembled, or complex:	Basic
Kinematic constraints:	None
Constraint moment output:	None
Available components:	ur_1, ur_2, ur_3
Kinetic moment output:	m_1, m_2, m_3
Orientation at a:	Required
Orientation at b:	Optional
Connector stops:	$\theta_1^{min} \le \alpha \le \theta_1^{max}$,

EULER	
	$\theta_2^{min} \le \beta \le \theta_2^{max}$,
	$\theta_3^{min} \le \gamma \le \theta_3^{max}$
Constitutive reference angles:	$\theta_1^{ref}, \theta_2^{ref}, \theta_3^{ref}$
Predefined friction parameters:	None
Contact force for predefined friction:	None

FLEXION-TORSION

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type FLEXION-TORSION provides a rotational connection between two nodes.
It models the bending and twisting of a cylindrical coupling between two shafts. In this case the response to twist rotations about the shafts may differ from the response to bending of the shafts. Connection type FLEXION-TORSION cannot be used in two-dimensional or axisymmetric analysis.

The flexural part of the connection resists angular misalignment of the two shafts, whereas the torsional part of the connection resists relative rotations about the shafts. Connection type FLEXION-TORSION can be used in conjunction with connection type RADIAL-THRUST when resistance to relative radial and thrust displacements is modeled.

Description

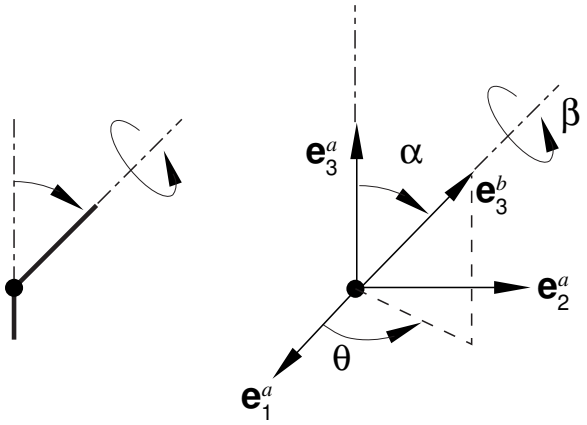


Figure 1: Connection type FLEXION-TORSION.

The FLEXION-TORSION connection does not impose kinematic constraints. The FLEXION-TORSION connection describes a finite rotation by three angles: flexion, torsion, and sweep (α, β, and θ). However, the flexion, torsion, and sweep angles do not represent three successive rotations. The flexion angle between two shafts measures the angle of misalignment of the two shafts and is always reported as a positive angle. The torsion angle measures the twist of one shaft relative to the other.

The sweep angle orients the rotation vector, in the $\mathbf{e}_1^a$–$\mathbf{e}_2^a$ plane, for the flexion motion. See *Figure 1*. Since the flexion angle is never negative, the sweep angle may undergo discontinuous jumps by up to π radians when the flexion angle passes through zero. An analysis may give inaccurate results or may not converge if any jump occurs in the sweep angle. In general, the sweep angle is not used as an available component of relative motion for which connector behavior is defined. Rather, it is used to define angular dependence for the elastic constitutive response in flexion deformations (as an independent component in the connector elastic behavior definition). Since the sweep angle is restricted to the

interval $-\pi$ to π radians, any dependence on the sweep angle should be periodic, such that the behavior for $\theta = -\pi$ is the same as $\theta = \pi$. Since $\alpha = 0$ is a singular point for which the sweep angle is not uniquely defined, it is strongly recommended that any connector behavior that defines flexural moment versus flexion angle gives zero moment at zero flexion angle. If connector behavior is defined in the sweep available component, the sweep moment must be zero at flexion angles $\alpha = 0$ and $\alpha = \pi$.

The FLEXION-TORSION connection is similar to a finite successive rotation parameterization 3–2–3. However, in terms of the 3–2–3 parameterization, the sweep angle is the first rotation angle, the flexion angle is the second rotation angle, and the torsion angle is the sum of the first and third rotation angles.

The first shaft direction at node *a* is $\mathbf{e}_3^a$, and the second shaft direction at node *b* is $\mathbf{e}_3^b$. Let the two shafts form an angle α, called the flexion angle. Then,

$$\alpha = \cos^{-1}\left(\mathbf{e}_3^a \cdot \mathbf{e}_3^b\right), \quad \text{where} \quad 0 \le \alpha \le \pi.$$

The flexion angle is a rotation by α about the (unit) rotation vector

$$\mathbf{q} = \frac{1}{\sin\alpha}\mathbf{e}_3^a \times \mathbf{e}_3^b, \quad \text{where} \quad \sin\alpha = \left\|\mathbf{e}_3^a \times \mathbf{e}_3^b\right\|.$$

The torsion angle β between the two shafts is defined as

$$\beta = \tan^{-1}\left(\frac{\mathbf{e}_2^a\cdot\mathbf{e}_1^b - \mathbf{e}_1^a\cdot\mathbf{e}_2^b}{\mathbf{e}_1^a\cdot\mathbf{e}_1^b + \mathbf{e}_2^a\cdot\mathbf{e}_2^b}\right) + m\pi,$$

where positive torsion angles are rotations about the positive $\mathbf{e}_3^b$-direction, and *m* is an integer.

The sweep angle θ measures the angle from $\mathbf{e}_1^a$ to the projection of $\mathbf{e}_3^b$ onto the $\mathbf{e}_1^a$–$\mathbf{e}_2^a$ plane. With this definition

$$\theta = \tan^{-1}\left(\frac{\mathbf{e}_2^a\cdot\mathbf{e}_3^b}{\mathbf{e}_1^a\cdot\mathbf{e}_3^b}\right), \quad \text{where} \quad -\pi \le \theta \le \pi.$$

It follows that the flexion rotation vector, $\mathbf{q}$, can be written

$$\mathbf{q} = -\sin\theta\mathbf{e}_1^a + \cos\theta\mathbf{e}_2^a.$$

A singularity in the definition of the sweep angles occurs when the flexion angle α vanishes. In this case $\mathbf{e}_3^b = \mathbf{e}_3^a$; that is, the torsion and sweep angle axes are coincident, and the two angles are no longer independent. When $\alpha = 0$, the sweep angle is assumed zero, $\theta = 0$.

The available components of relative motion ur_1, ur_2, and ur_3 are the changes in the flexion, torsion, and sweep angles and are defined as

$$ur_1 = \alpha - \alpha_0, \quad ur_2 = \beta - \beta_0, \quad \text{and} \quad ur_3 = \theta - \theta_0,$$

where α_0 and β_0 are the initial flexion and torsion angles, respectively. The initial value of the sweep angle θ_0 is chosen to be zero if the shafts align initially. The connector constitutive rotations are

$$ur_1^{mat} = \alpha - \theta_1^{ref}, \quad ur_2^{mat} = \beta - \theta_2^{ref}, \quad \text{and} \quad ur_3^{mat} = \theta - \theta_3^{ref}.$$

The kinetic moment in a FLEXION-TORSION connection is determined from the three component relationships:

$$m_1 = \mathbf{m}_{flex-tor}\cdot\mathbf{q}; \quad m_2 = \mathbf{m}_{flex-tor}\cdot\mathbf{e}_3^b; \quad \text{and} \quad m_3 = \mathbf{m}_{flex-tor}\cdot\mathbf{e}_3^a - \mathbf{m}_{flex-tor}\cdot\mathbf{e}_3^b.$$

Summary

FLEXION-TORSION	
Basic, assembled, or complex:	Basic
Kinematic constraints:	None
Constraint moment output:	None
Available components:	ur_1, ur_2, ur_3
Kinetic moment output:	m_1, m_2, m_3
Orientation at *a*:	Required
Orientation at *b*:	Optional
Connector stops:	$\theta_1^{min} \le \alpha \le \theta_1^{max}$,
	$\theta_2^{min} \le \beta \le \theta_2^{max}$,
	$\theta_3^{min} \le \theta \le \theta_3^{max}$
Constitutive reference angles:	$\theta_1^{ref}, \theta_2^{ref}, \theta_3^{ref}$
Predefined friction parameters:	None
Contact force for predefined friction:	None

FLOW-CONVERTER

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type FLOW-CONVERTER converts the relative rotation about a user-specified axis between the two nodes of the connector into material flow degree of freedom (10) at the second node of a connector element. This connection type can be used to model retractor and pretensioner devices in automotive seat belts (see *Seat belt analysis of a simplified crash dummy*) or cable drums in winch-like devices. Belt or cable material is considered to be wrapped around an axle or a drum, and material can be spooled either into or out of the connector element.

In certain cases, material flow needs to be converted into a displacement rather than a rotation. Examples include pretensioner devices for which experimental force vs. displacement data need to be specified. Although this connection type always converts the material flow into a rotation, the two modeling cases are equivalent. The experimentally available force vs. displacement data can be input directly as moment vs. rotation data for the same end result.

This connection type activates degree of freedom 10 at the second node of a connector. As with any other nodal degree of freedom, you must be careful in constraining it. This is typically done by attaching the connector to a SLIPRING connector that is part of the belt system or by applying a boundary condition. FLOW-CONVERTER connections cannot be used in two-dimensional and axisymmetric analyses in Abaqus/Explicit.

Description

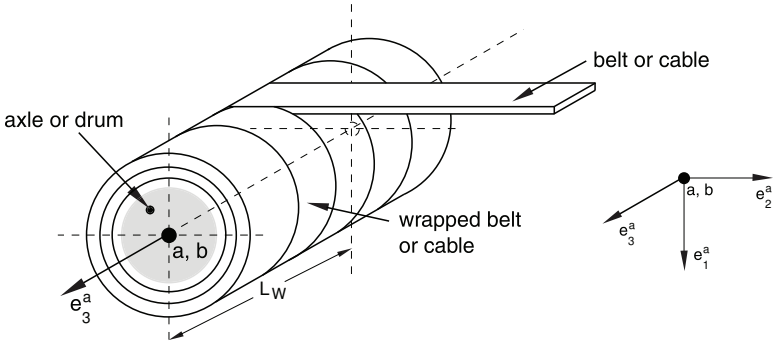


Figure 1: Connection type FLOW-CONVERTER.

The FLOW-CONVERTER connection constrains the relative rotation between the two nodes about the third local direction, $\mathbf{e}_3^a$, to the material flow at node b, Ψ_b. The constraint can be written as

$$ur_3 = \mathbf{e}_3^a \cdot (\theta_a - \theta_b) - \beta_s \Psi_b = 0,$$

where $\theta_a - \theta_b$ is the relative nodal rotation between node a and$b$ and β_s is a scaling factor specified as part of the associated connector section definition. By default, $\beta_s = 1.0$. The local direction $\mathbf{e}_3^a$ rotates with the nodal rotation at node a.

There are no available components of relative motion for this connection type; hence, kinetic behavior cannot be specified. However, the following kinematic quantities are available for output:

$$ur_1 = \mathbf{e}_3^a \cdot (\theta_a - \theta_b) \quad \text{and} \quad ur_2 = \Psi_b,$$

which will be output as CPR1 and CPR2, respectively.

The constraint moment is

$$\overline{\mathbf{m}} = m_3 \mathbf{e}_3^a \, .$$

Limitation

At most two FLOW-CONVERTER connectors can share their second node where degree of freedom 10 is active.

Summary

FLOW-CONVERTER	
Basic, assembled, or complex:	Specialized basic rotational
Kinematic constraints:	$ur_3 = 0$
Constraint moment output:	m_3
Available components:	None
Kinetic force output:	None
Orientation at *a*:	Required
Orientation at *b*:	Ignored
Connector stops:	None
Constitutive reference lengths:	None
Predefined friction parameters:	None
Contact force for predefined friction:	None

HINGE

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type HINGE joins the position of two nodes and provides a revolute constraint between their rotational degrees of freedom.
Connection type HINGE cannot be used in two-dimensional or axisymmetric analysis.

Description

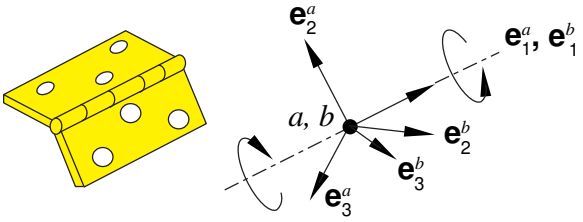


Figure 1: Connection type HINGE.

Connection type HINGE imposes kinematic constraints and uses local orientation definitions equivalent to combining connection types JOIN and REVOLUTE.

The connector constraint forces and moments reported as connector output depend strongly on the order and the location of the nodes in the connector element (see *Connector Behavior*). Since the kinematic constraints are enforced at node *b* (the second node of the connector element), the reported forces and moments are the constraint forces and moments applied at node *b* to enforce the HINGE constraint. Thus, in most cases the connector output associated with a HINGE connection is best interpreted when node *b* is located at the center of the device enforcing the constraint. This choice is essential when moment-based friction is modeled in the connector since the contact forces are derived from the connector forces and moments, as illustrated below. Proper enforcement of the kinematic constraints is independent of the order or location of the nodes.

Friction

Predefined Coulomb-like friction in the HINGE connection relates the kinematic constraint forces and moments in the connector to a friction moment (CSM1) in the rotation about the hinge axis. The table below summarizes the parameters that are used to specify predefined friction in this connection type as discussed in detail next. A typical interpretation of the geometric scaling constants is illustrated in *Figure 2*.

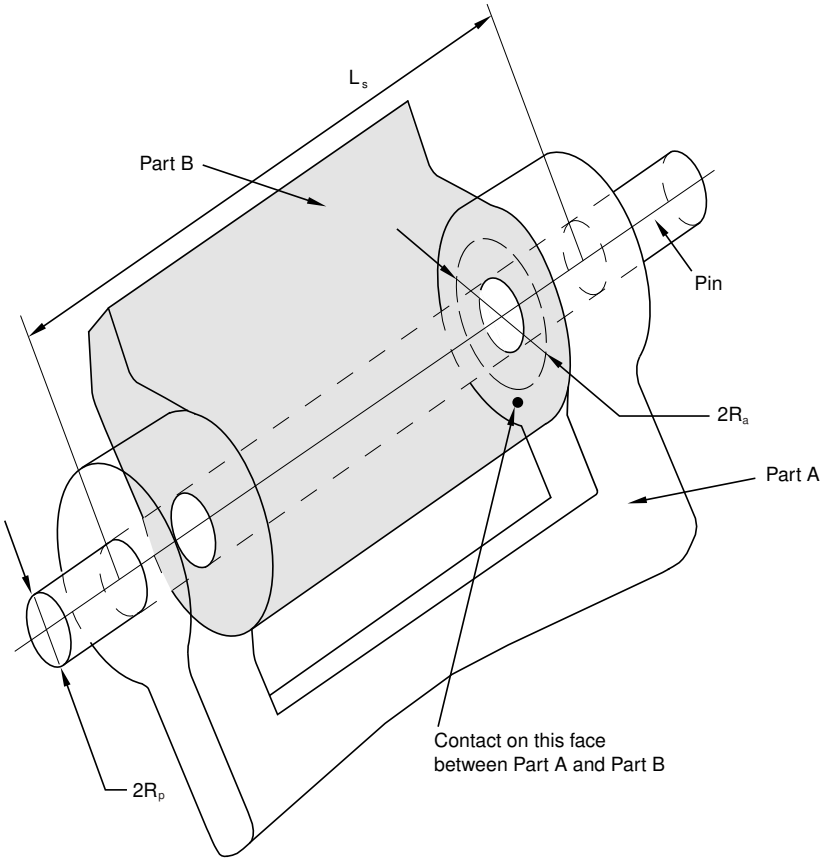


Figure 2: Illustration of the geometric scaling constants for a HINGE connection.

Since the rotation about the 1-direction is the only possible relative motion in the connection, the frictional effect is formally written in terms of moments generated by tangential tractions and moments generated by contact forces, as follows:

$$\Phi = \mathrm{P}(\mathbf{f}) - \mu \mathrm{M_N} \leq 0,$$

where the potential $\mathrm{P}(\mathbf{f})$ represents the moment magnitude of the frictional tangential tractions in the connector in a direction tangent to the cylindrical surface on which contact occurs, $\mathrm{M_N}$ is the friction-producing normal moment on the same cylindrical surface, and μ is the friction coefficient. Frictional stick occurs if $\Phi < 0$; and sliding occurs if $\Phi = 0$, in which case the friction moment is $\mu \mathrm{M_N}$.

The normal moment $\mathrm{M_N}$ is the sum of a magnitude measure of friction-producing connector moments, $\mathrm{M_C} = g(\mathbf{f})$, and a self-equilibrated internal contact moment (such as from a press-fit assembly), $\mathrm{M_C^{int}}$:

$$\mathrm{M_N} = \left| \mathrm{M_C} + \mathrm{M_C^{int}} \right| = \left| g(\mathbf{f}) + \mathrm{M_C^{int}} \right|.$$

The magnitude measure of friction-producing connector contact moments, $\mathrm{M_C}$, is defined by summing the following contributions:

- a moment from an axial force, $F_a R_a$, where $F_a = |f_1|$ and R_a is an effective friction arm associated with the constraint force in the axial direction (the R_a radius could be interpreted as an average radius of the outer sleeve cylindrical sections as found in a typical door hinge or as an effective radius associated with the hinge end caps, if they exist; if R_a is 0.0, F_a is ignored); and

- a moment from normal forces to the cylindrical face, $F_n R_p$, where R_p is the radius of the pin cross-section in the local 2–3 plane and F_n is itself a sum of the following two contributions:
  - a radial force contribution, F_r (the magnitude of the constraint forces enforcing the translation constraints in the local 2–3 plane):

$$F_r = \sqrt{f_2^2 + f_3^2}, \text{and}$$

  - a force contribution from "bending," F_{bend}, obtained by scaling the bending moment, M_{bend} (the magnitude of the constraint moments enforcing the REVOLUTE constraint), by a length factor, as follows:

$$M_{bend} = \sqrt{m_2^2 + m_3^2},$$

$$F_{bend} = 2\frac{M_{bend}}{L_s},$$

    where L_s represents a characteristic overlapping length between the pin and the sleeve. If L_s is 0.0, M_{bend} is ignored.

Thus,

$$\mathrm{M_C} = g(\mathbf{f}) = F_a R_a + F_n R_p = |f_1 R_a| + \sqrt{(R_p f_2)^2 + (R_p f_3)^2} + \sqrt{(\beta m_2)^2 + (\beta m_3)^2},$$

where $\beta = \frac{2R_p}{L_s}$.

The moment magnitude of the frictional tangential tractions, $\mathbf{P}(\mathbf{f}) = |m_1|$.

Summary

HINGE	
Basic, assembled, or complex:	Assembled
Kinematic constraints:	JOIN + REVOLUTE
Constraint force and moment output:	f_1, f_2, f_3, m_2, m_3
Available components:	ur_1
Kinetic force and moment output:	m_1
Orientation at *a*:	Required
Orientation at *b*:	Optional
Connector stops:	$\theta_1^{min} \le \alpha \le \theta_1^{max}$
Constitutive reference lengths:	θ_1^{ref}
Predefined friction parameters:	Required: R_p; optional: R_a, L_s, $\mathrm{M_C^{int}}$
Contact moment for predefined friction:	$\mathrm{M_C}$

JOIN

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type JOIN makes the position of two nodes the same.
If the two nodes are not colocated initially, the position of node b is fixed relative to that of node a in a Cartesian coordinate system attached to node a.

Even though an orientation is optional at node a, connection type JOIN does not activate rotational degrees of freedom at node a.

Description

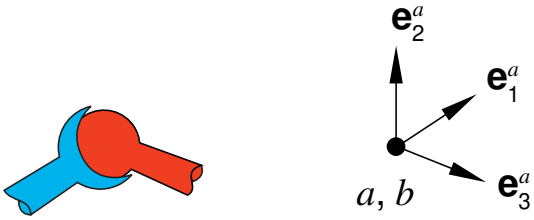


Figure 1: Connection type JOIN.

The JOIN connection makes the position of node b equal to that of node a. If the two nodes are not coincident initially, the Cartesian coordinates of node b relative to node a are fixed. See connection type CARTESIAN for a definition of the Cartesian coordinates of node b relative to node a. If rotational degrees of freedom exist at node a, the local directions corotate with the node.

The constraint force in the JOIN connection acts in the three local directions at node a and is

$$\bar{\mathbf{f}} = f_1\mathbf{e}_1^a + f_2\mathbf{e}_2^a + f_3\mathbf{e}_3^a ,$$

where $f_3 = 0$ in two-dimensional analysis.

Friction

When used by itself, there is no predefined Coulomb-like friction in the JOIN connection, since there are no available components of relative motion for which friction can be defined. However, when the JOIN and REVOLUTE connection types are used together, the predefined friction is the same as the HINGE connection. When the JOIN and UNIVERSAL connection types are used together, the predefined friction is the same as the UJOINT connection.

Summary

JOIN	
Basic, assembled, or complex:	Basic
Kinematic constraints:	$u_1 = 0, u_2 = 0, u_3 = 0$
Constraint force output:	f_1, f_2, f_3
Available components:	None
Kinetic force output:	None
Orientation at *a*:	Optional
Orientation at *b*:	Ignored
Connector stops:	None
Constitutive reference lengths:	None
Predefined friction parameters:	None
Contact force for predefined friction:	None

LINK

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type LINK maintains a constant distance between two nodes.
Rotational degrees of freedom, if they exist, are not affected at either node.

Description

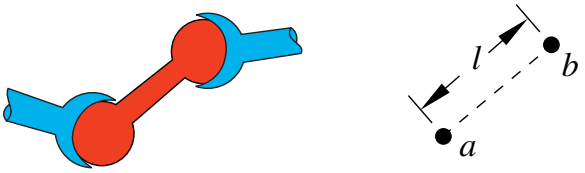


Figure 1: Connection type LINK.

The LINK connection constrains the position of node b, $\mathbf{x}_b$, to a constant distance from node a. The distance between the two nodes is

$$l = \|\mathbf{x}_b - \mathbf{x}_a\|$$

and is constant. The constraint force in the LINK connection acts along the line connecting the two nodes and is

$$\bar{\mathbf{f}} = f_1 \mathbf{q}\,, \quad \text{where} \quad \mathbf{q} = \frac{1}{\|\mathbf{x}_b - \mathbf{x}_a\|} (\mathbf{x}_b - \mathbf{x}_a)\,.$$

Summary

LINK	
Basic, assembled, or complex:	Basic
Kinematic constraints:	$l = constant$
Constraint force output:	f_1
Available components:	None
Kinetic force output:	None
Orientation at *a*:	Ignored
Orientation at *b*:	Ignored
Connector stops:	None

LINK	
Constitutive reference lengths:	None
Predefined friction parameters:	None
Contact force for predefined friction:	None

PLANAR

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type PLANAR provides a local two-dimensional system in a three-dimensional analysis. Connection type PLANAR cannot be used in two-dimensional or axisymmetric analysis.

Description

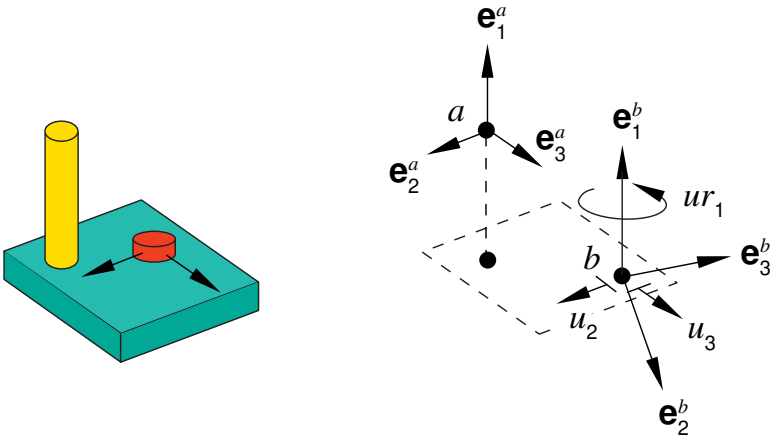


Figure 1: Connection type PLANAR.

Connection type PLANAR imposes kinematic constraints and uses local orientation definitions equivalent to combining connection types SLIDE-PLANE and REVOLUTE.

Friction

Predefined Coulomb-like friction in the PLANAR connection relates the kinematic constraint forces and moments in the connector to the friction forces in the translations in the local 2–3 plane and the frictional moment in the rotation about the local 1-direction. These two frictional effects are discussed separately below.

1. The frictional effect due to sliding in the 2–3 plane is formally written as

$$\Phi_C = P_C(\mathbf{f}) - \mu F_{N_C} \leq 0,$$

where the potential $P_C(\mathbf{f})$ represents the magnitude of the frictional tangential tractions in the connector in a direction tangent to the local 2–3 plane on which contact occurs, F_{N_C} is the friction-producing normal force on the same plane, and μ is the friction coefficient. Frictional stick occurs if $\Phi_C < 0$; and sliding occurs if $\Phi_C = 0$, in which case the friction force (CSFC) is μF_{N_C}.

The normal force $\mathbf{F}_{N_C}$ is the sum of a magnitude measure of force-producing connector forces, $\mathbf{F}_C = g(\mathbf{f})$, and a self-equilibrated internal contact force, $\mathbf{F}_C^{int}$:

$$\mathbf{F}_{N_C} = \left|\mathbf{F}_C + \mathbf{F}_C^{int}\right| = \left|g(\mathbf{f}) + \mathbf{F}_C^{int}\right|.$$

The contact force magnitude $\mathbf{F}_C$ is defined by summing the following two contributions:

- a force contribution, $F_1 = |f_1|$ (the constraint force enforcing the SLIDE-PLANE constraint); and
- a force contribution from "bending," F_{bend}, obtained by scaling the bending moment, M_{bend} (the magnitude of the constraint moments enforcing the REVOLUTE constraint), by a length factor, as follows:

$$M_{bend} = \sqrt{m_2^2 + m_3^2},$$

$$F_{bend} = \frac{M_{bend}}{R},$$

where R represents a characteristic radius of the "puck" (as illustrated in *Figure 2*) in the local 2–3 plane. If R is 0.0, M_{bend} is ignored.

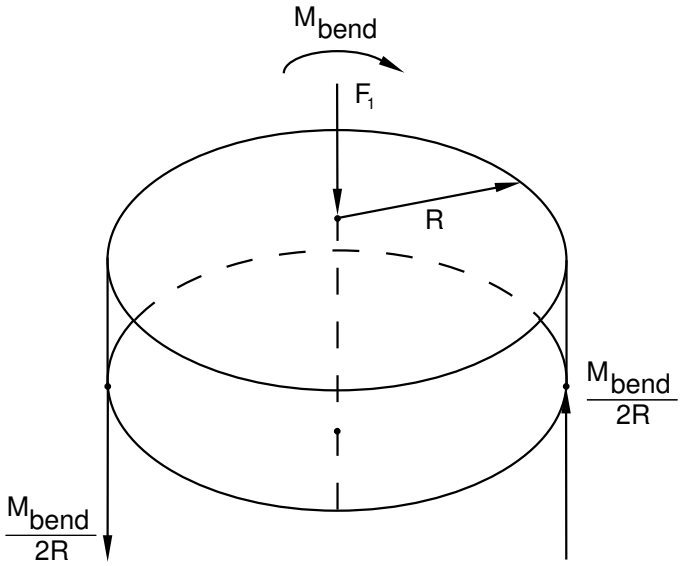


Figure 2: Illustration of the effective internal friction contact forces.

Thus,

$$\mathbf{F}_C = g(\mathbf{f}) = F_1 + F_{bend} = |f_1| + \sqrt{(\beta m_2)^2 + (\beta m_3)^2},$$

where $\beta = \frac{1}{R}$.

The magnitude of the frictional tangential moment, $\mathbf{P}_C(\mathbf{f})$, is computed using

$$\mathbf{P}_C(\mathbf{f}) = \sqrt{f_2^2 + f_3^2}.$$

2. Since the frictional effects due to rotation about the 1-direction are quantified, the frictional effect is formally written in terms of moments generated by tangential tractions and moments generated by contact forces as

$$\Phi_{R1} = \mathbf{P}_{R1}(\mathbf{f}) - \mu \mathbf{M}_{N_{R1}} \leq 0,$$

where the potential $\mathbf{P}_{\mathbf{R1}}(\mathbf{f})$ represents the magnitude of the frictional tangential moment in the connector about the 1-direction, $\mathbf{M}_{\mathbf{N}_{\mathbf{R1}}}$ is the friction-producing normal moment about the same axis, and μ is the friction coefficient. Frictional stick in rotation occurs if $\Phi_{\mathbf{R1}} < \mathbf{0}$; and sliding occurs if $\Phi_{\mathbf{R1}} = \mathbf{0}$, in which case the friction moment (CSM1) is $\mu\mathbf{M}_{\mathbf{N}_{\mathbf{R1}}}$.

The normal moment $\mathbf{M}_{\mathbf{N}_{\mathbf{R1}}}$ is the sum of a magnitude measure of friction-producing connector moments, $\mathbf{M}_{\mathbf{R1}} = g(\mathbf{f})$, and a self-equilibrated internal contact moment, $\mathbf{M}_{\mathbf{R1}}^{\mathbf{int}}$:

$$\mathbf{M}_{\mathbf{N}_{\mathbf{R1}}} = \left|\mathbf{M}_{\mathbf{R1}} + \mathbf{M}_{\mathbf{R1}}^{\mathbf{int}}\right| = \left|g(\mathbf{f}) + \mathbf{M}_{\mathbf{R1}}^{\mathbf{int}}\right|.$$

The contact moment magnitude $\mathbf{M}_{\mathbf{R1}}$ is defined by summing the following two contributions:

- a moment from a contact force in the 2–3 plane, M_1 (the constraint moment enforcing the SLIDE-PLANE constraint):

  $$M_1 = \frac{2}{3}F_1 R,$$

  where $F_1 = |f_1|$, R represents a characteristic radius of the “puck” (as illustrated in *Figure 2*) in the local 2–3 plane (if R is 0.0, M_1 is ignored), and the 2/3 factor comes from integrating moment contributions from a uniform pressure ($\frac{F_1}{\pi R^2}$) over the circular contact patch; and
- a moment contribution from “bending,” M_{bend} (the magnitude of the constraint moments enforcing the REVOLUTE constraint):

  $$M_{bend} = \sqrt{{m_2}^2 + {m_3}^2}.$$

Thus,

$$\mathbf{M}_{\mathbf{R1}} = g(\mathbf{f}) = M_1 + M_{bend} = \frac{2}{3}R|f_1| + \sqrt{m_2^2 + m_3^2}.$$

The magnitude of the frictional tangential tractions, $\mathbf{P}_{\mathbf{R1}}(\mathbf{f})$, is computed using

$$\mathbf{P}_{\mathbf{R1}}(\mathbf{f}) = |m_1|.$$

Summary

PLANAR	
Basic, assembled, or complex:	Assembled
Kinematic constraints:	SLIDE-PLANE + REVOLUTE
Constraint force and moment output:	f_1, m_2, m_3
Available components:	u_2, u_3, ur_1
Kinetic force and moment output:	f_2, f_3, m_1
Orientation at *a*:	Required
Orientation at *b*:	Optional

PLANAR	
Connector stops:	$l_2^{min} \le y \le l_2^{max}$,
	$l_3^{min} \le z \le l_3^{max}$,
	$\theta_1^{min} \le \alpha \le \theta_1^{max}$
Constitutive reference lengths and angles:	$l_2^{ref}, l_3^{ref}, \theta_1^{ref}$
Predefined friction parameters:	Optional: R, $\mathbf{F}_{\mathbf{C}}^{\mathbf{int}}$, $\mathbf{M}_{\mathbf{R1}}^{\mathbf{int}}$
Contact forces and moments for predefined friction:	$\mathbf{F}_{\mathbf{C}}$, $\mathbf{M}_{\mathbf{R1}}$

PROJECTION CARTESIAN

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type PROJECTION CARTESIAN provides a connection between two nodes where the response in three local connection directions (that is, the axes of the local Cartesian coordinate system) is measured.
Unlike the CARTESIAN connection, which uses an orthonormal coordinate system that follows node *a*, the PROJECTION CARTESIAN connection uses an orthonormal system that follows the systems at both nodes *a* and *b*.

The connector local directions used in the PROJECTION CARTESIAN connection are identical to those used in the PROJECTION FLEXION-TORSION connection. Connection type PROJECTION CARTESIAN is compatible with connection type PROJECTION FLEXION-TORSION and is appropriate for modeling the displacement response of bushing-like or spot-weld-like components.

Description

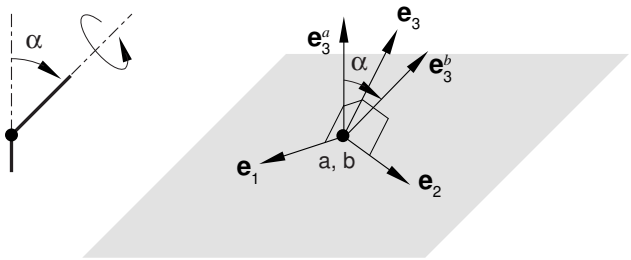


Figure 1: Connection type PROJECTION CARTESIAN.

The PROJECTION CARTESIAN connection does not impose kinematic constraints. It defines three local directions $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ as a function of the directions at both nodes *a* and *b*. These directions are the projection directions defined by the PROJECTION FLEXION-TORSION connection. The PROJECTION CARTESIAN connection measures the change in position of node *b* relative to node *a* along the (projection) coordinate directions $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$.

The position of node *b* relative to node *a* is

$$x = \mathbf{e}_1 \cdot (\mathbf{x}_b - \mathbf{x}_a) \ ; \quad y = \mathbf{e}_2 \cdot (\mathbf{x}_b - \mathbf{x}_a) \ ; \quad \text{and} \quad z = \mathbf{e}_3 \cdot (\mathbf{x}_b - \mathbf{x}_a) \ .$$

The available components of relative motion are

$$u_1 = x - x_0 \ ; \quad u_2 = y - y_0 \ ; \quad \text{and} \quad u_3 = z - z_0 \ ,$$

where x_0, y_0, and z_0 are the initial coordinates of node *b* relative to node *a* along the initial $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ directions. The connector constitutive displacements are

$$u_1^{mat} = x - l_1^{ref} \ ; \quad u_2^{mat} = y - l_2^{ref} \ ; \quad \text{and} \quad u_3^{mat} = z - l_3^{ref} .$$

The local directions in a PROJECTION CARTESIAN connection are “centered” between the systems at the two connector nodes. PROJECTION CARTESIAN connections are appropriate where isotropic or anisotropic material response is modeled and the local material directions evolve as a function of the rotations at both ends of the connection. The kinetic force is

$$\mathbf{f}_{projCart} = f_1\mathbf{e}_1 + f_2\mathbf{e}_2 + f_3\mathbf{e}_3 \, .$$

In two-dimensional analysis $z = 0$, $u_3 = 0$, $u_3^{mat} = 0$, and $f_3 = 0$.

Summary

PROJECTION CARTESIAN	
Basic, assembled, or complex:	Basic
Kinematic constraints:	None
Constraint force output:	None
Available components:	u_1, u_2, u_3
Kinetic force output:	f_1, f_2, f_3
Orientation at *a*:	Optional
Orientation at *b*:	Optional
Connector stops:	$l_1^{min} \le x \le l_1^{max}$,
	$l_2^{min} \le y \le l_2^{max}$,
	$l_3^{min} \le z \le l_3^{max}$
Constitutive reference lengths:	$l_1^{ref}, l_2^{ref}, l_3^{ref}$
Predefined friction parameters:	None
Contact force for predefined friction:	None

PROJECTION FLEXION-TORSION

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type PROJECTION FLEXION-TORSION provides a rotational connection between two nodes. It models the bending and twisting of a cylindrical coupling between two shafts. In this case the response to twist rotations about the shafts may differ from the response to bending of the shafts. Connection type PROJECTION FLEXION-TORSION is similar to connection type FLEXION-TORSION. Whereas the FLEXION-TORSION connection has rotation parameterization angles consisting of total flexion, torsion, and sweep, the PROJECTION FLEXION-TORSION connection has rotation parameterization angles consisting of two component flexion angles and a torsion angle. The flexion angle of the FLEXION-TORSION connection is the resultant flexion angle resulting from the two component flexion angles of the PROJECTION FLEXION-TORSION connection. Connection type PROJECTION FLEXION-TORSION cannot be used in two-dimensional or axisymmetric analysis.

The flexural part of the connection resists angular misalignment of the two shafts, whereas the torsional part of the connection resists relative rotations about the shafts. Connection type PROJECTION FLEXION-TORSION can be used in conjunction with connection type PROJECTION CARTESIAN when modeling the response of bushing-like or spot-weld-like components.

Description

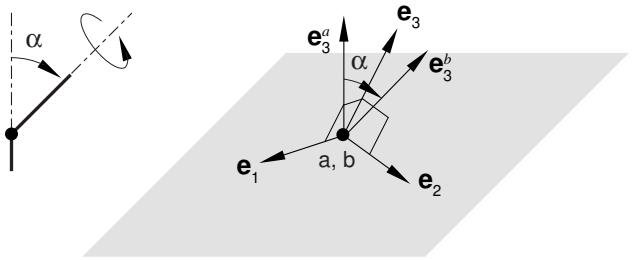


Figure 1: Connection type PROJECTION FLEXION-TORSION.

The PROJECTION FLEXION-TORSION connection does not impose kinematic constraints. The PROJECTION FLEXION-TORSION connection describes a finite rotation by three angles: flexion 1, flexion 2, and torsion ($\boldsymbol{\alpha_1}$, $\boldsymbol{\alpha_2}$, and $\boldsymbol{\beta}$). However, the flexion 1, flexion 2, and torsion angles do not represent three successive rotations. The two component flexion angles ($\boldsymbol{\alpha_1}$ and $\boldsymbol{\alpha_2}$) make up the total flexion angle between two shafts and measure the angle of misalignment of the two shafts. The torsion angle measures the twist of one shaft relative to the other.

The first shaft direction at node *a* is $\mathbf{e}_3^a$, and the second shaft direction at node *b* is $\mathbf{e}_3^b$. Let the two shafts form an angle $\boldsymbol{\alpha}$, called the total flexion angle. Then,

$$\alpha = \cos^{-1}\left(\mathbf{e}_3^a \cdot \mathbf{e}_3^b\right), \quad \text{where} \quad 0 \le \alpha \le \pi.$$

The flexion angle is a rotation by α about the (unit) rotation vector,

$$\mathbf{q} = \frac{1}{\sin\alpha}\mathbf{e}_3^a \times \mathbf{e}_3^b \ , \quad \text{where} \quad \sin\alpha = \left\|\mathbf{e}_3^a \times \mathbf{e}_3^b\right\| .$$

The PROJECTION FLEXION-TORSION connection is formulated in terms of the unit vector normal to a plane, $\mathbf{e}_3$, and two unit vectors spanning this plane, $\mathbf{e}_1$ and $\mathbf{e}_2$. See *Figure 1*. The plane with normal vector $\mathbf{e}_3$ is referred to as the flexion-torsion plane. The component flexion angles α_1 and α_2 are determined from α and $\mathbf{q}$ by projection onto the two in-plane directions:

$$\alpha_1 = \alpha\left(\mathbf{e}_1 \cdot \mathbf{q}\right) \quad \text{and} \quad \alpha_2 = \alpha\left(\mathbf{e}_2 \cdot \mathbf{q}\right) .$$

The torsion angle in a PROJECTION FLEXION-TORSION connection can be understood from a finite successive rotation parameterization 3–2–3. In terms of the 3–2–3 parameterization the total flexion angle is the second successive rotation angle, and the torsion angle is the sum of the first and third successive rotation angles. The torsion angle β between the two shafts is defined as

$$\beta = \tan^{-1}\left(\frac{\mathbf{e}_2^a\cdot\mathbf{e}_1^b - \mathbf{e}_1^a\cdot\mathbf{e}_2^b}{\mathbf{e}_1^a\cdot\mathbf{e}_1^b + \mathbf{e}_2^a\cdot\mathbf{e}_2^b}\right) + m\pi ,$$

where positive torsion angles are rotations about the positive $\mathbf{e}_3$-direction and m is an integer.

The PROJECTION FLEXION-TORSION connection avoids the singularity that occurs in the sweep angle of the FLEXION-TORSION connection when the total flexion angle α vanishes. As a result, the PROJECTION FLEXION-TORSION connection is better suited for defining bushing-like behavior for flexion response that varies with the direction of $\mathbf{q}$ in the flexion-torsion plane.

The available components of relative motion ur_1, ur_2, and ur_3 are the changes in the two flexion angles and the torsion angle and are defined as

$$ur_1 = \alpha_1 - \alpha_{10} \ , \quad ur_2 = \alpha_2 - \alpha_{20} \ , \quad \text{and} \quad ur_3 = \beta - \beta_0 ,$$

where α_{10}, α_{20}, and β_0 are the initial flexion component angles and torsion angle, respectively. The connector constitutive rotations are

$$ur_1^{mat} = \alpha_1 - \theta_1^{ref} \ , \quad ur_2^{mat} = \alpha_2 - \theta_2^{ref} \ , \quad \text{and} \quad ur_3^{mat} = \beta - \theta_3^{ref} .$$

The kinetic moment in a PROJECTION FLEXION-TORSION connection is

$$\mathbf{m}_{projflex-tor} = m_1\mathbf{e}_1 + m_2\mathbf{e}_2 + m_3\mathbf{e}_3 .$$

Summary

PROJECTION FLEXION-TORSION	
Basic, assembled, or complex:	Basic
Kinematic constraints:	None
Constraint moment output:	None
Available components:	ur_1, ur_2, ur_3
Kinetic moment output:	m_1, m_2, m_3
Orientation at *a*:	Required
Orientation at *b*:	Optional
Connector stops:	$\theta_1^{min} \le \alpha_1 \le \theta_1^{max}$,

PROJECTION FLEXION-TORSION	
	$\theta_2^{min} \leq \alpha_2 \leq \theta_2^{max}$,
	$\theta_3^{min} \leq \beta \leq \theta_3^{max}$
Constitutive reference angles:	$\theta_1^{ref}, \theta_2^{ref}, \theta_3^{ref}$
Predefined friction parameters:	None
Contact force for predefined friction:	None

RADIAL-THRUST

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type RADIAL-THRUST provides a connection between two nodes where the response differs in the radial and cylindrical axis directions.
Connection type RADIAL-THRUST models situations such as a point inside a cylindrical bearing where the response to radial displacements differs from the response to thrusting motions. Connection type RADIAL-THRUST cannot be used in two-dimensional or axisymmetric analysis.

If the rotational degrees of freedom at the two nodes are connected through flexural and torsional resistance, connection type FLEXION-TORSION can be used in conjunction with connection type RADIAL-THRUST.

Description

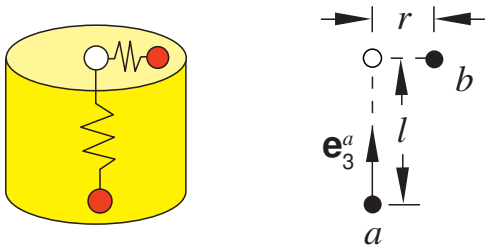


Figure 1: Connection type RADIAL-THRUST.

The RADIAL-THRUST connection does not impose kinematic constraints. An orientation at node a is required to define the axis of the rectangular coordinate system, $\mathbf{e}_3^a$. The position of node b relative to node a is given by the radial and axial-direction distances

$$r = \sqrt{[\mathbf{e}_1^a \cdot (\mathbf{x}_b - \mathbf{x}_a)]^2 + [\mathbf{e}_2^a \cdot (\mathbf{x}_b - \mathbf{x}_a)]^2} \quad \text{and} \quad l = \mathbf{e}_3^a \cdot (\mathbf{x}_b - \mathbf{x}_a) \ .$$

The RADIAL-THRUST connection has two available components of relative motion, u_1 and u_3. The radial displacement u_1 measures the change in distance from node b to the axis of the cylindrical coordinate system and is defined as

$$u_1 = r - r_0 \ ,$$

where r_0 is the initial radial distance from node b to the axis. The thrust displacement u_3 measures the change in distance from node a to node b along the cylindrical axis and is defined as

$$u_3 = l - l_0 \ ,$$

where l_0 is the initial distance along the axis from node b to node a. The connector constitutive displacements are

$$u_1^{mat} = r - l_1^{ref} \quad \text{and} \quad u_3^{mat} = l - l_3^{ref} .$$

The kinetic force is

$$\mathbf{f}_{rad-thr} = f_1 \mathbf{e}_r + f_3 \mathbf{e}_3^a ,$$

where the radial unit vector is

$$\mathbf{e}_r = \frac{1}{r} \left[\mathbf{e}_1^a \cdot (\mathbf{x}_b - \mathbf{x}_a) \, \mathbf{e}_1^a + \mathbf{e}_2^a \cdot (\mathbf{x}_b - \mathbf{x}_a) \, \mathbf{e}_2^a \right] .$$

The radial resistance of the RADIAL-THRUST connector is analogous to a single spring in the $\mathbf{e}_1^a$–$\mathbf{e}_2^a$ plane. Loads applied in this plane and perpendicular to the current radial unit vector will initially encounter no resistance and may lead to numerical singularity and/or zero pivot warnings from the solver during static analyses. If the numerical singularities cause convergence difficulties, one modeling option is to overlay the RADIAL-THRUST connector with a CARTESIAN connector with a very small elastic stiffness.

Summary

RADIAL-THRUST	
Basic, assembled, or complex:	Basic
Kinematic constraints:	None
Constraint force output:	None
Available components:	u_1, u_3
Kinetic force output:	f_1, f_3
Orientation at *a*:	Required
Orientation at *b*:	Ignored
Connector stops:	$l_1^{min} \le r \le l_1^{max}$,
	$l_3^{min} \le l \le l_3^{max}$
Constitutive reference lengths:	l_1^{ref}, l_3^{ref}
Predefined friction parameters:	None
Contact force for predefined friction:	None

RETRACTOR

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type RETRACTOR joins the position of two nodes and provides a FLOW-CONVERTER constraint between the material flow degree of freedom (10) at the second node and the rotational degrees of freedom at the first node of the connector.
This connection type can be used to model retractor and pretensioner devices in automotive seat belts (see *Seat belt analysis of a simplified crash dummy*) or cable drums in winch-like devices.

RETRACTOR connections cannot be used in two-dimensional and axisymmetric analyses in Abaqus/Explicit.

Description

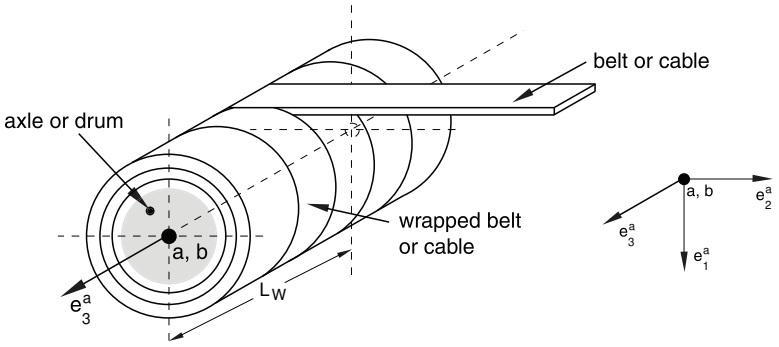


Figure 1: Connection type RETRACTOR.

Connection type RETRACTOR imposes kinematic constraints and uses local orientation definitions equivalent to combining connection types JOIN and FLOW-CONVERTER.

Summary

RETRACTOR	
Basic, assembled, or complex:	Assembled
Kinematic constraints:	JOIN + FLOW-CONVERTER
Constraint force output:	f_1, f_2, f_3, m_3
Available components:	None
Kinetic force output:	None
Orientation at *a*:	Required

RETRACTOR	
Orientation at *b*:	Ignored
Connector stops:	None
Constitutive reference lengths:	None
Predefined friction parameters:	None
Contact force for predefined friction:	None

REVOLUTE

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type REVOLUTE provides a connection between two nodes where the rotations are constrained about two local directions and free about a shared axis.
The shared axis of rotation is the connector local 1-direction. Connection type REVOLUTE cannot be used in two-dimensional or axisymmetric analysis.

Connection type REVOLUTE models the rotational part of a HINGE or CYLINDRICAL joint.

Description

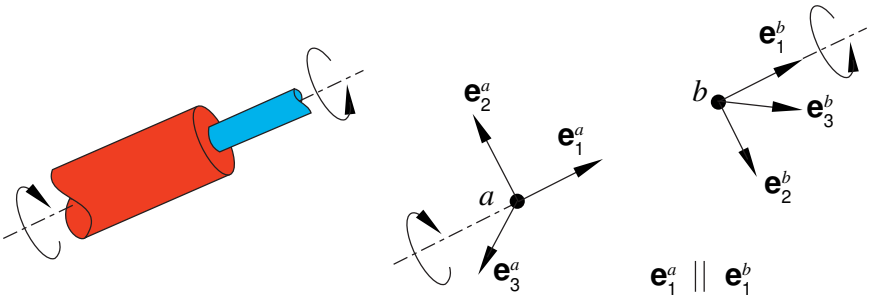


Figure 1: Connection type REVOLUTE.

A REVOLUTE connection constrains two rotational components of relative motion between two nodes and allows one free rotational component. The two kinematic constraints imposed by the REVOLUTE connection are

$$\mathbf{e}_1^a \cdot \mathbf{e}_2^b = 0 \quad \text{and} \quad \mathbf{e}_1^a \cdot \mathbf{e}_3^b = 0 ,$$

which are equivalent to the requirement that $\mathbf{e}_1^a = \mathbf{e}_1^b$. Alternatively, the REVOLUTE constraint is equivalent to setting the second and third Cardan angles to zero in a CARDAN connection. If the shared axes $\mathbf{e}_1^a$ and $\mathbf{e}_1^b$ do not align initially, the REVOLUTE constraint will hold the second and third Cardan angles fixed at their initial values. The constraint moment in the REVOLUTE connection is

$$\overline{\mathbf{m}} = m_2 \mathbf{e}_2^a + m_3 \mathbf{e}_3^a .$$

Node b can rotate about the shared local direction $\mathbf{e}_1^a = \mathbf{e}_1^b$. The relative angular position of the local directions at node b relative to a is

$$\alpha = -\tan^{-1}\left(\frac{\mathbf{e}_2^a \cdot \mathbf{e}_3^b}{\mathbf{e}_3^a \cdot \mathbf{e}_3^b}\right) ,$$

where α is the first Cardan angle measuring a counterclockwise rotation about the $\mathbf{e}_1^a$-direction of $\mathbf{e}_2^a$ to $\mathbf{e}_2^b$.

The available component of relative motion, ur_1, measures the change in angular position and is defined as

$$ur_1 = \alpha - \alpha_0 + n\pi\,,$$

where α_0 is the initial angular position and *n* is an integer accounting for multiple rotations about the shared axis. The connector constitutive rotation is

$$ur_1^{mat} = \alpha - \theta_1^{ref} + n\pi\,.$$

The kinetic moment in the REVOLUTE connection is

$$\mathbf{m}_{revolute} = m_1 \mathbf{e}_1^a\,.$$

Friction

When used by itself, there is no predefined Coulomb-like friction in the REVOLUTE connection. However, when the REVOLUTE connection is used in combination with a JOIN, SLIDE-PLANE, or SLOT connection, the predefined friction is the same as the HINGE, PLANAR, and CYLINDRICAL connections, respectively.

Summary

REVOLUTE	
Basic, assembled, or complex:	Basic
Kinematic constraints:	$\mathbf{e}_1^a \cdot \mathbf{e}_2^b = 0, \mathbf{e}_1^a \cdot \mathbf{e}_3^b = 0$
Constraint moment output:	m_2, m_3
Available components:	ur_1
Kinetic moment output:	m_1
Orientation at *a*:	Required
Orientation at *b*:	Optional
Connector stops:	$\theta_1^{min} \le \alpha \le \theta_1^{max}$
Constitutive reference angles:	θ_1^{ref}
Predefined friction parameters:	None
Contact moment for predefined friction:	None

ROTATION

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type ROTATION provides a rotational connection between two nodes where the relative rotation between the nodes is parameterized by the rotation vector.
In two-dimensional and axisymmetric analyses, the ROTATION connection type involves a single (scalar) relative rotation component.

Although available components of relative motion exist for the ROTATION connection type in three-dimensional analysis, the finite rotation parameterization of the connection is not necessarily well-suited for defining connector behavior. If a finite, three-dimensional ROTATION connection with connector behavior is desired, either the CARDAN or EULER connection type typically is more appropriate.

When connection type ROTATION is used in a connector element connected to ground at the element's first node, the rotational components relative to the orientation at ground are identical to the Abaqus convention for nodal rotation degrees of freedom. Hence, connection type ROTATION can be used in conjunction with prescribed connector motion (see *Connector Actuation*) to specify finite rotation boundary conditions in local coordinate directions using the Abaqus convention for finite rotation boundary conditions.

Description

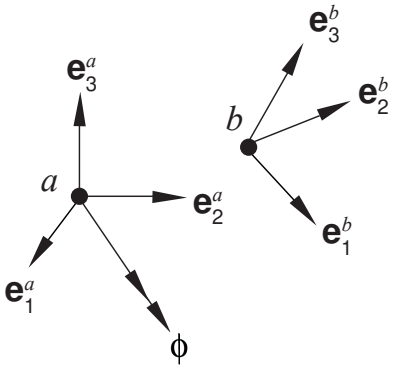


Figure 1: Connection type ROTATION.

The rotation connection does not impose kinematic constraints. The rotation connection is a finite rotation connection where the local directions at node b are parameterized relative to the local directions at node a by the rotation vector.

Let $\boldsymbol{\phi}$ be the rotation vector that positions local directions $\{\mathbf{e}_1^b, \mathbf{e}_2^b, \mathbf{e}_3^b\}$ relative to $\{\mathbf{e}_1^a, \mathbf{e}_2^a, \mathbf{e}_3^a\}$; that is,

$$\mathbf{e}_i^b = \exp\left[\widehat{\boldsymbol{\phi}}\right] \cdot \mathbf{e}_i^a$$

for all $i = 1, 2, 3$, where $\widehat{\phi}$ is the skew-symmetric matrix with axial vector ϕ. See *Rotation variables* for a discussion of finite rotations.

The available components of relative motion in the ROTATION connection are the change in the rotation vector components positioning the local directions at node *b* relative to the local directions at node *a*. Therefore,

$$ur_i = \phi_i - (\phi_0)_i + 2n\pi\frac{\phi_i}{\|\phi\|},$$

where ϕ_0 is the initial rotation vector, $n \geq 0$ is an integer accounting for rotations with magnitude greater than 2π, all vector components are components relative to the local directions $\mathbf{e}_i^a$, and $i = 1, 2, 3$. The connector constitutive rotations are

$$ur_i^{mat} = \phi_i - \theta_i^{ref} + 2n\pi\frac{\phi_i}{\|\phi\|}.$$

The kinetic moment in a rotation connection is

$$m_i = \mathbf{m}_{rotation} \cdot \mathbf{e}_i^a, \quad i = 1, 2, 3.$$

In two-dimensional and axisymmetric analyses $ur_1 = ur_2 = 0$ and $m_1 = m_2 = 0$.

Summary

ROTATION	
Basic, assembled, or complex:	Basic
Kinematic constraints:	None
Constraint moment output:	None
Available components:	ur_1, ur_2, ur_3
Kinetic moment output:	m_1, m_2, m_3
Orientation at *a*:	Optional
Orientation at *b*:	Optional
Connector stops:	$\theta_i^{min} \leq \phi_i \leq \theta_i^{max}$
Constitutive reference angles:	θ_i^{ref}
Predefined friction parameters:	None
Contact force for predefined friction:	None

ROTATION-ACCELEROMETER

Products: Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type ROTATION-ACCELEROMETER provides a convenient way to measure the relative angular position, velocity, and acceleration of a body in a local coordinate system.
These kinematic quantities are measured relative to the motion of node *a* and are reported in the coordinate system of node *b*. Each node of the connector can translate and rotate independently, although fixing the first of the two nodes to ground is more common. With the first node fixed, connection type ROTATION-ACCELEROMETER provides a convenient way to measure the local components of the angular velocity and angular acceleration in a coordinate system fixed to a moving body (for example, an accelerometer).

Connection type ROTATION-ACCELEROMETER is available only in Abaqus/Explicit. It is the rotation counterpart to connection type ACCELEROMETER, which measures relative translational position, velocity, and acceleration.

ROTATION-ACCELEROMETER connectors cannot be used in two-dimensional and axisymmetric analysis in Abaqus/Explicit.

Description

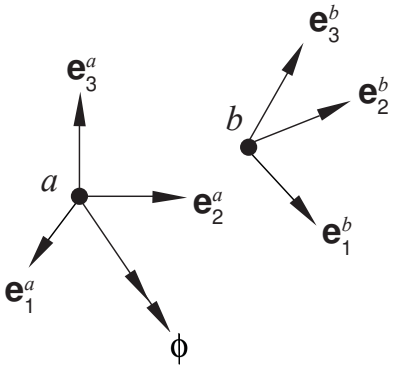


Figure 1: Connection type ROTATION-ACCELEROMETER.

The ROTATION-ACCELEROMETER connection does not impose kinematic constraints. It defines three local directions at node *a* and three local directions at node *b*. The ROTATION-ACCELEROMETER connection's formulation is similar to that for the ROTATION connection. The ROTATION-ACCELEROMETER connection measures the finite rotation that takes the local directions at node *a* into the local directions at node *b* and parameterizes that finite rotation by the rotation vector. Let $\boldsymbol{\phi}$ be the rotation vector that positions local directions $\left\{\mathbf{e}_1^b, \mathbf{e}_2^b, \mathbf{e}_3^b\right\}$ relative to $\left\{\mathbf{e}_1^a, \mathbf{e}_2^a, \mathbf{e}_3^a\right\}$; that is,

$$\mathbf{e}_i^b = \mathbf{exp}\left[\widehat{\boldsymbol{\phi}}\right] \cdot \mathbf{e}_i^a$$

for all $i = 1, 2, 3$, where $\widehat{\phi}$ is the skew-symmetric matrix with axial vector ϕ. See *Rotation variables* for a discussion of finite rotations. The connection measures the change in the rotation vector components in the local directions rotating with the body at node *b*. The rotation vector components are calculated as

$$\phi_i = \mathbf{e}_i^b \cdot \phi \,.$$

There are no available components of relative motion for the ROTATION-ACCELEROMETER connection. The connector rotation is

$$ur_i = \phi_i - (\phi_0)_i + 2n\pi \frac{\phi_i}{\|\phi\|} \,,$$

where ϕ_0 is the initial rotation vector and $n \geq 0$ is an integer accounting for rotations with magnitude greater than 2π.

The ROTATION-ACCELEROMETER connection differs from the ROTATION connection in the way angular velocity and acceleration are calculated. The ROTATION-ACCELEROMETER connection measures velocity and acceleration from the nodes as

$$vr_i = (\omega_b - \omega_a) \cdot \mathbf{e}_i^b \quad \text{and} \quad ar_i = (\alpha_b - \alpha_a) \cdot \mathbf{e}_i^b \,,$$

where ω_a, ω_b, α_a, and α_b are the nodal angular velocities and accelerations at nodes *a* and *b*, respectively.

In two-dimensional and axisymmetric analyses $ur_1 = ur_2 = 0$.

Summary

ROTATION-ACCELEROMETER	
Basic, assembled, or complex:	Basic
Kinematic constraints:	None
Constraint force output:	None
Available components:	None
Kinetic force output:	None
Orientation at *a*:	Optional
Orientation at *b*:	Optional
Connector stops:	None
Constitutive reference lengths:	None
Predefined friction parameters:	None
Contact force for predefined friction:	None

SLIDE-PLANE

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type SLIDE-PLANE keeps node b on a plane defined by the orientation of node a and the initial position of node b.
Connection type SLIDE-PLANE cannot be used in two-dimensional or axisymmetric analysis. The normal direction defining the plane at node a is $\mathbf{e}_1^a$.

Connection type SLIDE-PLANE models a point confined between parallel plates or a pin-in-slot connection where the pin is free to move normal to the plane of the slot.

Description

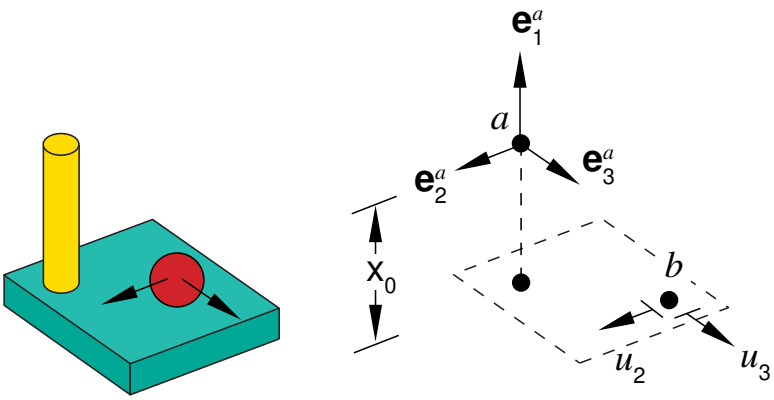


Figure 1: Connection type SLIDE-PLANE.

The SLIDE-PLANE connection constrains the position of node b, $\mathbf{x}_b$, to remain on a plane defined by the local normal direction $\mathbf{e}_1^a$. The normal direction distance from node a to the plane is constant:

$$x = \mathbf{e}_1^a \cdot (\mathbf{x}_b - \mathbf{x}_a) = x_0 \,,$$

where x_0 is the initial distance from node a to the plane. The constraint force in the SLIDE-PLANE connection is

$$\bar{\mathbf{f}} = f_1 \mathbf{e}_1^a \,.$$

Node b can move in the plane defined by the normal of node a. The position of node b in the plane relative to node a is

$$y = \mathbf{e}_2^a \cdot (\mathbf{x}_b - \mathbf{x}_a) \quad \text{and} \quad z = \mathbf{e}_3^a \cdot (\mathbf{x}_b - \mathbf{x}_a) \,.$$

The two available components of relative motion, u_2 and u_3, are

$$u_2 = y - y_0 \quad \text{and} \quad u_3 = z - z_0 \,,$$

where y_0 and z_0 are the coordinates of the initial position of node *b*. The connector constitutive displacements are

$$u_2^{mat} = y - l_2^{ref} \quad \text{and} \quad u_3^{mat} = z - l_3^{ref} \,.$$

The kinetic force in the plane is

$$\mathbf{f}_{s-p} = f_2 \mathbf{e}_2^a + f_3 \mathbf{e}_3^a \,.$$

Friction

Predefined Coulomb-like friction in the SLIDE-PLANE connection relates the kinematic constraint forces in the connector to the friction forces (CSFC) in the translations along the two local directions in the 2–3 plane.

The frictional effect is formally written as

$$\Phi = \mathrm{P}(\mathbf{f}) - \mu \mathrm{F_N} \leq 0,$$

where the potential $\mathrm{P}(\mathbf{f})$ represents the magnitude of the frictional tangential tractions in the connector in a direction tangent to the 2–3 plane on which contact occurs, $\mathrm{F_N}$ is the friction-producing normal force on the same plane, and μ is the friction coefficient. Frictional stick occurs if $\Phi < 0$; and sliding occurs if $\Phi = 0$, in which case the friction force is $\mu \mathrm{F_N}$.

The normal force $\mathrm{F_N}$ is the sum of a magnitude measure of friction-producing connector forces, $\mathrm{F_C} = g(\mathbf{f})$, and a self-equilibrated internal contact force, $\mathrm{F_C^{int}}$:

$$\mathrm{F_N} = \left| \mathrm{F_C} + \mathrm{F_C^{int}} \right| = \left| g(\mathbf{f}) + \mathrm{F_C^{int}} \right| .$$

The force magnitude $\mathrm{F_C} = |f_1|$.

The magnitude of the frictional tangential tractions, $\mathrm{P}(\mathbf{f})$, is computed using

$$\mathrm{P}(\mathbf{f}) = \sqrt{f_2^2 + f_3^2}.$$

The predefined Coulomb-like friction is computed differently when the SLIDE-PLANE connection is used in combination with a REVOLUTE connection. See the description of the PLANAR connection for the predefined friction definition in this case.

Summary

SLIDE-PLANE	
Basic, assembled, or complex:	Basic
Kinematic constraints:	$x = x_0$
Constraint force output:	f_1
Available components:	u_2, u_3
Kinetic force output:	f_2, f_3
Orientation at *a*:	Required
Orientation at *b*:	Ignored

SLIDE-PLANE	
Connector stops:	$l_2^{min} \le y \le l_2^{max}$,
	$l_3^{min} \le z \le l_3^{max}$
Constitutive reference lengths:	l_2^{ref}, l_3^{ref}
Predefined friction parameters:	Optional: $\mathbf{F}_C^{int}$
Contact force for predefined friction:	$\mathbf{F}_C$

SLIPRING

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type SLIPRING provides a connection between two nodes that models material flow and stretching between two points of a belt system.

It can be used to model seat belts (see *Seat belt analysis of a simplified crash dummy*), pulley systems, and taut cable systems.

The angle between two adjacent belt segments is used only for friction calculations. By default, Abaqus/Explicit computes the angle, α, automatically from the nodal coordinates as an angle between 0 and π. By default, Abaqus/Standard, assumes the angle, α, to be 0 throughout the analysis. You can override the default value and specify the angle between two adjacent belt segments (in radians) as part of the connector section definition, which allows you to specify wrapping angles larger than π. Alternatively, in Abaqus/Standard you can request that the angle be computed automatically, based upon the nodal coordinates, as part of the connector section definition. Abaqus/Standard uses the nodal coordinates from the beginning of an increment to compute the contact angle.

This connection type activates the material flow degree of freedom (10) at both nodes of the connector. As with any other nodal degree of freedom, you must be careful in constraining it. This is typically done by attaching the connector to other SLIPRING connectors that are part of the belt system, attaching it to a RETRACTOR (FLOW-CONVERTER) connector, or applying a boundary condition.

SLIPRING connections cannot be used in two-dimensional and axisymmetric analyses in Abaqus/Explicit.

Description

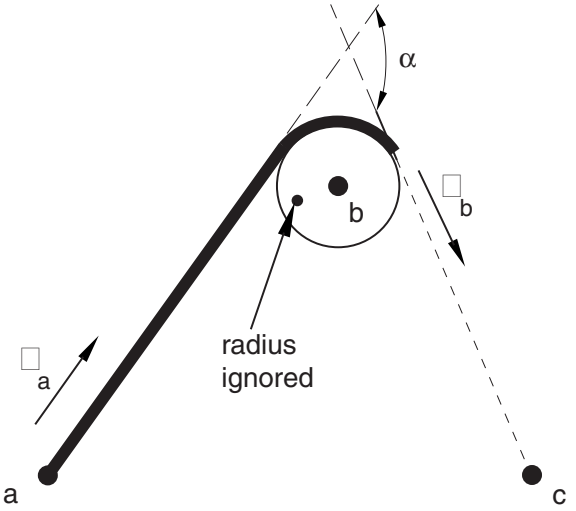


Figure 1: Connection type SLIPRING.

The SLIPRING connection does not constrain any component of relative motion. Hence, there is no restriction on the position of the connector nodes.

The distance between nodes is

$$d_{ab} = \|\mathbf{x}_b - \mathbf{x}_a\| .$$

The belt material can flow and stretch between nodes *a* and *b*. Flow can occur with no stretching (such as in a rigid belt), stretching can occur with no flow (such as when the flow is constrained at both nodes of the connector), or both flow and stretching can occur simultaneously (such as in compliant belts). By convention, the material flow at node *a* is positive if it enters segment ab and is positive at node *b* if it exits the segment. A reference length can be defined in incremental fashion as

$$l^{ref}_{new} = l^{ref}_{old} + \Delta\Psi_a - \Delta\Psi_b,$$

where l^{ref}_{new} is the reference length at the end of the current increment, l^{ref}_{old} is the reference length at the beginning of the current increment, $\Delta\Psi_a$ is the incremental flow at node *a*, and $\Delta\Psi_b$ is the incremental flow at node *b*. The stretch in the belt can then be defined as

$$d = \frac{d_{ab}}{l^{ref}_{new}},$$

and the “strain” in the belt can be computed as

$$u_1 = u^{mat}_1 = d - 1.$$

At the beginning of the analysis, the reference length at $t = 0$ is

$$l^{ref}\big|_{t=0} = \frac{d_{ab}|_{t=0}}{d_p},$$

where d_p is the initial stretch of the belt. By default, the initial stretch is $d_p = 1.0$ meaning that there are no initial strains in the belt. You can specify initial strains in the belt, $u_1|_{t=0}$, by specifying a connector constitutive reference. The initial stretch is then computed using

$$d_p = u_1|_{t=0} + 1.$$

The second available component of relative motion is simply the material flow past node *b*,

$$u_2 = u^{mat}_2 = \Psi_b.$$

The third component of relative motion is the material flow into node *a* and is used only for output:

$$u_3 = u^{mat}_3 = \Psi_a.$$

The kinetic force is

$$\mathbf{f}_{slipring} = f_1 l^{ref}_{new}\mathbf{q} \ , \quad \text{where} \quad \mathbf{q} = \frac{1}{\|\mathbf{x}_b - \mathbf{x}_a\|}\left(\mathbf{x}_b - \mathbf{x}_a\right) .$$

Limitations

At most two SLIPRING connectors can share a common node. The following limitations apply with respect to the kinetic behavior that can be defined in the SLIPRING connection type:

- Only predefined friction can be defined in the second component of relative motion as outlined below.
- In Abaqus/Explicit plasticity, damage and lock connector behavior cannot be specified.

- The connectivities of the two adjacent SLIPRING connector elements sharing a common node *b* (*Figure 1*) should be in the typical order *a*–*b* and *b*–*c*. In addition, any two adjacent SLIPRING connector elements must refer to the same connector behavior except for the friction data.

Friction

Predefined Coulomb-like friction in the SLIPRING connection relates the tension in the belt segment ab (kinetic force f_1 in component 1) to the tension in the adjacent belt segment bc. In the simpler case of frictionless sliding, the two tensions are equal (apart from inertial effects due to the motion of the belt in dynamic analyses). If frictional effects are included as material flows past node *b*, the two tensions differ by the total friction force (CSF2) over the contact arch between the belt and the ring (angle α).

The Coulomb-like frictional effect is a well-known analytical result. In the case when frictional sliding occurs in the direction illustrated in *Figure 1*, the tensions in the two segments, $f_{ab} = f_1$ and f_{bc}, are related as follows:

$$f_{ab} = f_{bc} e^{-\mu\alpha},$$

where μ is the friction coefficient. The friction force is simply the difference

$$CSF2 = f_{bc} - f_{ab}.$$

More formally, the frictional relationship is modeled by considering the potential function

$$\Phi = f_{bc} e^{-\mu\alpha} - f_{ab}.$$

Frictional stick occurs if $\Phi < 0$; and sliding occurs if $\Phi = 0$, in which case the tension force $f_{ab} = f_{bc} e^{-\mu\alpha}$. Friction forces do not develop if the kinetic force f_1 is compressive. When sliding occurs in the opposite direction, the sign of the exponent in the potential equation changes.

The friction force is reported as f_2 in this connection type. The friction-generating “contact force” is reported as CNF2=f_1.

An Abaqus/Standard analysis might experience convergence difficulties, including reverse slip, if the ratio of the specified stick stiffness (see *Defining the Stick Stiffness*) to the belt elastic stiffness (normalized by belt length) is very large. To alleviate such difficulties, ensure that the ratio is moderate (less than 1000). If you use the default stick stiffness, you can alleviate convergence difficulties by increasing the slip tolerance (see *Controlling the Allowable Elastic Slip as a Fraction of the Characteristic Facet Dimension*).

In Abaqus/Explicit, by default, the distance between the two nodes of the SLIPRING is not allowed to become less then one hundredth of the original distance between the nodes, which prevents the SLIPRING from collapsing to zero length during the analysis. The two nodes of the SLIPRING can move apart after coming to the minimum distance configuration during the analysis. In addition, the belt can continue to slip over the nodes while they are stopped at the minimum distance configuration. This default value of the minimum distance can be overridden by specifying a lower limit of the connector stop in component 1 of the SLIPRING.

Output

Some of the connector output variables have a somewhat different meaning for this connection type than usual, as follows:

- CP1 is the current distance between the nodes;
- CP2 is the material flow at node *b*;
- CP3 is the material flow at node *a*; and
- CU1 is the strain (dimensionless) in the segment ab.

Summary

SLIPRING	
Basic, assembled, or complex:	Complex
Kinematic constraints:	None
Constraint force output:	None
Available components:	u_1, u_2, u_3
Kinetic force output:	f_1, f_2
Orientation at *a*:	Ignored
Orientation at *b*:	Ignored
Connector stops:	None
Constitutive reference lengths:	$d_p - 1$
Predefined friction parameters:	None
Contact force for predefined friction:	f_1

SLOT

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type SLOT provides a connection where node b stays on the line defined by the orientation of node a and the initial position of node b.

The line of action of the slot is the $\mathbf{e}_1^a$-direction.

In three-dimensional analysis node b cannot move in the direction normal to the slot; i.e., the $\mathbf{e}_3^a$ direction. If node b is free to move in the normal direction, connection type SLIDE-PLANE should be used.

Description

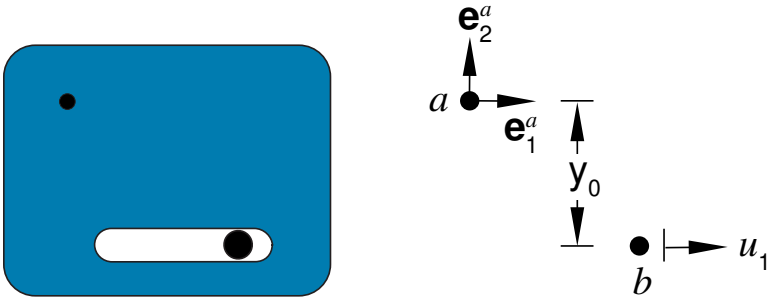


Figure 1: Connection type SLOT.

The line of the slot is defined by the first local direction at node a, $\mathbf{e}_1^a$, and the initial position of node b. The SLOT connection constrains the position of node b, $\mathbf{x}_b$, to remain on the line of the slot. Therefore, the relative position of node b is fixed in the directions perpendicular to the slot:

$$y = \mathbf{e}_2^a \cdot (\mathbf{x}_b - \mathbf{x}_a) = y_0 \,,$$

where y_0 is the initial distance from node a to the slot in the local 2-direction. In three dimensions

$$z = \mathbf{e}_3^a \cdot (\mathbf{x}_b - \mathbf{x}_a) = z_0 \,,$$

where z_0 is the initial distance from node a to the slot in the local 3-direction. The constraint force in the slot is

$$\bar{\mathbf{f}} = f_2 \mathbf{e}_2^a + f_3 \mathbf{e}_3^a \,,$$

where $f_3 = 0$ in two-dimensional analysis.

Node b can move along the line of the slot. The relative position in the slot is the distance between node b and node a along the $\mathbf{e}_1^a$-direction and is defined as

$$x = \mathbf{e}_1^a \cdot (\mathbf{x}_b - \mathbf{x}_a) \ .$$

The available component of relative motion is the displacement u_1, which measures the change of the relative position in length along the slot and is defined as

$$u_1 = x - x_0 \ ,$$

where x_0 is the initial distance between node *b* and node *a* along the slot. The connector constitutive displacement is

$$u_1^{mat} = x - l_1^{ref} \ .$$

The kinetic force in the slot is

$$\mathbf{f}_{slot} = f_1 \mathbf{e}_1^a \ .$$

Friction

Predefined Coulomb-like friction in the SLOT connection relates the kinematic constraint forces in the connector to the friction force (CSF1) in the translation along the slot.

The frictional effect is formally written as

$$\Phi = \mathbf{P}(\mathbf{f}) - \mu \mathbf{F}_\mathbf{N} \leq 0,$$

where the potential $\mathbf{P}(\mathbf{f})$ represents the magnitude of the frictional tangential tractions in the connector in a direction tangent to the slot axis along which contact occurs, $\mathbf{F}_\mathbf{N}$ is the friction-producing normal (contact) force in the direction normal to the slot, and μ is the friction coefficient. Frictional stick occurs if $\Phi < 0$; and sliding occurs if $\Phi = 0$, in which case the friction force is $\mu \mathbf{F}_\mathbf{N}$.

The normal force $\mathbf{F}_\mathbf{N}$ is the sum of a magnitude measure of the friction-producing connector force, $\mathbf{F}_\mathbf{C} = g(\mathbf{f})$, and a self-equilibrated internal contact force, $\mathbf{F}_\mathbf{C}^\mathbf{int}$:

$$\mathbf{F}_\mathbf{N} = \left| \mathbf{F}_\mathbf{C} + \mathbf{F}_\mathbf{C}^\mathbf{int} \right| = \left| g(\mathbf{f}) + \mathbf{F}_\mathbf{C}^\mathbf{int} \right| .$$

The force magnitude $\mathbf{F}_\mathbf{C}$ is computed using

$$\mathbf{F}_\mathbf{C} = \sqrt{f_2^2 + f_3^2}.$$

The magnitude of the frictional tangential tractions $\mathbf{P}(\mathbf{f}) = |f_1|$.

The predefined Coulomb-like friction is computed differently when the SLOT connection is used in combination with a REVOLUTE or an ALIGN connection. See CYLINDRICAL and TRANSLATOR, respectively, for the predefined friction definition in these cases.

Summary

SLOT	
Basic, assembled, or complex:	Basic
Kinematic constraints:	$y = y_0, \ z = z_0$
Constraint force output:	f_2, f_3
Available components:	u_1

SLOT	
Kinetic force output:	f_1
Orientation at *a*:	Required
Orientation at *b*:	Ignored
Connector stops:	$l_1^{min} \le l \le l_1^{max}$
Constitutive reference lengths:	l_1^{ref}
Predefined friction parameters:	Optional: $\mathbf{F}_C^{int}$
Contact force for predefined friction:	$\mathbf{F}_C$

TRANSLATOR

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type TRANSLATOR provides a slot constraint between two nodes and aligns their local directions.

Description

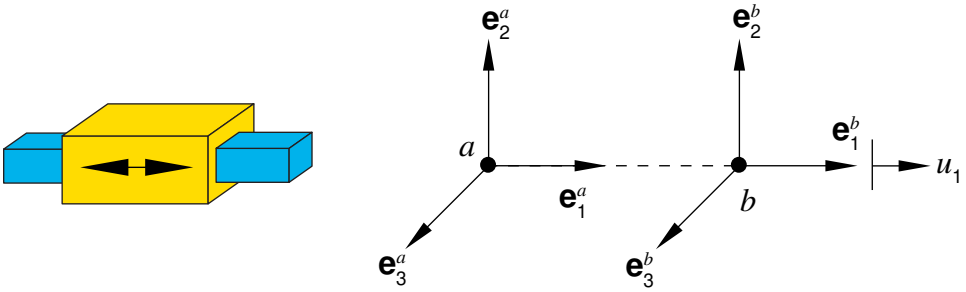


Figure 1: Connection type TRANSLATOR.

Connection type TRANSLATOR imposes kinematic constraints and uses local orientation definitions equivalent to combining connection types SLOT and ALIGN.

The connector constraint forces and moments reported as connector output depend strongly on the order and location of the nodes in the connector (see *Connector Behavior*). Since the kinematic constraints are enforced at node *b* (the second node of the connector element), the reported forces and moments are the constraint forces and moments applied at node *b* to enforce the TRANSLATOR constraint. Thus, in most cases the connector output associated with a TRANSLATOR connection is best interpreted when node *b* is located at the center of the device enforcing the constraint. This choice is essential when moment-based friction is modeled in the connector since the contact forces are derived from the connector forces and moments, as illustrated below. Proper enforcement of the kinematic constraints is independent of the order or location of the nodes.

Friction

Predefined Coulomb-like friction in the TRANSLATOR connection relates the kinematic constraint forces and moments in the connector to the friction force (CSF1) in the translation along the slot.

The frictional effect is formally written as

$$\Phi = \mathbf{P}(\mathbf{f}) - \mu \mathbf{F_N} \leq 0,$$

where the potential $\mathbf{P}(\mathbf{f})$ represents the magnitude of the frictional tangential traction in the connector in the local 1-direction, $\mathbf{F_N}$ is the friction-producing normal (contact) force in the direction normal to the slot, and μ is the friction coefficient. Frictional stick occurs if $\Phi < 0$; and sliding occurs if $\Phi = 0$, in which case the friction force is $\mu \mathbf{F_N}$.

The normal force $\mathbf{F_N}$ is the sum of a magnitude measure of contact friction-producing connector forces, $\mathbf{F_C} = g(\mathbf{f})$, and a self-equilibrated internal contact force, $\mathbf{F_C^{int}}$:

$$\mathbf{F_N} = \left|\mathbf{F_C} + \mathbf{F_C^{int}}\right| = \left|g(\mathbf{f}) + \mathbf{F_C^{int}}\right|.$$

The contact force magnitude $\mathbf{F_C}$ is defined by summing the following three contributions:

- a force contribution from torque, F_{torq}, obtained by scaling the torque constraint moment about the 1-direction, M_{torq}, by a length factor, as follows:

  $$M_{torq} = |m_1|,$$
  $$F_{torq} = \frac{M_{torq}}{R_r},$$

  where R_r represents the effective radius of the shaft cross-section in the local 2–3 plane (if R_r is 0.0, M_{torq} is ignored);
- a radial force contribution, F_r (the magnitude of the constraint forces enforcing the SLOT constraint):

  $$F_r = \sqrt{{f_2}^2 + {f_3}^2};$$

  and
- a force contribution from "bending," F_{bend}, obtained by scaling the bending constraint moment, M_{bend}, by a length factor, as follows:

  $$M_{bend} = \sqrt{{m_2}^2 + {m_3}^2},$$
  $$F_{bend} = 2\frac{M_{bend}}{L},$$

  where L represents a characteristic overlapping length in the slot direction. If L is 0.0, M_{bend} is ignored.

Thus,

$$\mathbf{F_C} = g(\mathbf{f}) = F_{torq} + F_r + F_{bend} = \left|\frac{m_1}{R_r}\right| + \sqrt{{f_2}^2 + {f_3}^2} + \sqrt{(\beta m_2)^2 + (\beta m_3)^2},$$

where $\beta = \frac{2}{L}$.

The magnitude of the frictional tangential tractions, $\mathbf{P}(\mathbf{f})$, is $|f_1|$.

Summary

TRANSLATOR	
Basic, assembled, or complex:	Assembled
Kinematic constraints:	SLOT + ALIGN
Constraint force and moment output:	f_2, f_3, m_1, m_2, m_3
Available components:	u_1
Kinetic force and moment output:	f_1

TRANSLATOR	
Orientation at *a*:	Required
Orientation at *b*:	Optional
Connector stops:	$l_1^{min} \le l \le l_1^{max}$
Constitutive reference lengths:	l_1^{ref}
Predefined friction parameters:	Optional: R_r, L, $\mathbf{F}_C^{int}$
Contact force for predefined friction:	$\mathbf{F}_C$

UJOINT

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type UJOINT joins the position of two nodes and provides a universal constraint between their rotational degrees of freedom.
Connection type UJOINT cannot be used in two-dimensional or axisymmetric analysis.

Description

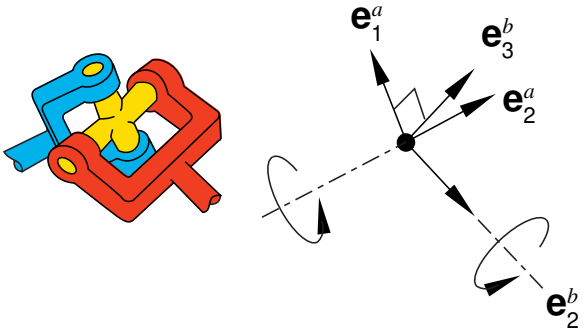


Figure 1: Connection type UJOINT.

Connection type UJOINT imposes kinematic constraints and uses local orientation definitions equivalent to combining connection types JOIN and UNIVERSAL.

The connector constraint forces and moments reported as connector output depend strongly on the order of the nodes and location of the nodes in the connector (see *Connector Behavior*). Since the kinematic constraints are enforced at node *b* (the second node of the connector element), the reported forces and moments are the constraint forces and moments applied at node *b* to enforce the UJOINT constraint. Thus, in most cases the connector output associated with a UJOINT connection is best interpreted when node *b* is located at the center of the device enforcing the constraint. This choice is essential when moment-based friction is modeled in the connector since the contact forces are derived from the connector forces and moments, as illustrated below. Proper enforcement of the kinematic constraints is independent of the order or location of the nodes.

Friction

Predefined Coulomb-like friction in the UJOINT connection relates the kinematic constraint forces and moments in the connector to friction moments about the unconstrained rotations (about the two directions of the connection cross). The UJOINT connection type consists of four hinge-like connections placed at the four ends of the connection cross (see *Figure 1*) that generate frictional moments about the cross axes. The frictional moments in each of these hinges are computed in a fashion similar to the HINGE connection.

The constraint forces and moments are used first to compute a reaction force, F_r (the magnitude of the constraint forces enforcing the JOIN constraint), and a "twisting" constraint moment, M_{twist} (the magnitude of the constraint moment enforcing the UNIVERSAL connection), as follows:

$$F_r = \sqrt{f_1^2 + f_2^2 + f_3^2},$$
$$M_{twist} = |m_2|.$$

The two cross directions are given by $\mathbf{e}_1^a$ and $\mathbf{e}_3^b$. The constraint moment, M_{twist}, acts about an axis perpendicular to the connection cross given by $\mathbf{e}_{cross} = \mathbf{e}_1^a \times \mathbf{e}_3^b$. Both F_r and M_{twist} are considered to be applied at the center of the connection cross. The constraint moment, M_{twist}, produces in each of the four hinges a bending-like moment about $\mathbf{e}_{cross}$:

$$M_{twist}^{hinge} = \alpha_{twist} M_{twist}$$

and a transverse force in the cross plane

$$F_{twist}^{hinge} = \beta_{twist} \frac{M_{twist}}{L_a},$$

where L_a represents a characteristic length of the cross arm between the center of the cross and the ends of the cross. The scaling factors α_{twist} and β_{twist} are nonlinear functions of the slenderness of the cross axes (the aspect ratio L_a/R_p, where R_p is the average radius of the four pins at the ends of the connection cross): they can be approximated by assuming the cross arm with rigid bodies for infinitely small aspect ratios, with Timoshenko beams for small aspect ratios (less than 20), and with Euler-Bernoulli beams for slender axes (large aspect ratios). Abaqus chooses the appropriate values automatically based on the user-specified geometric constants L_a and R_p. *Figure 2* illustrates the evolution of the scaling factors as a function of the aspect ratio: as the aspect ratio approaches 0.0, α_{twist} approaches 0.0 and β_{twist} approaches 0.25; for large aspect ratios, α_{twist} approaches 0.125 and β_{twist} approaches 0.375.

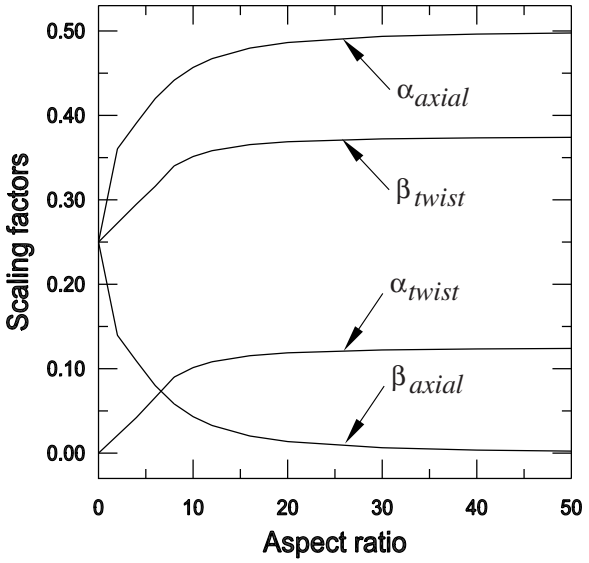


Figure 2: Scaling factors in the UJOINT connection.

The constraint force, F_r, can be decomposed into axial forces along the two axes of the connection cross and a "bending" force perpendicular to the connection cross plane:

$$F_{axial1} = \mathbf{F}_r \cdot \mathbf{e}_1^a,$$
$$F_{axial3} = \mathbf{F}_r \cdot \mathbf{e}_3^b, \text{and}$$
$$F_{bend} = \mathbf{F}_r \cdot \mathbf{e}_{cross},$$

where

$$\mathbf{F}_r = f_1 \mathbf{e}_1^a + f_2 \mathbf{e}_2^a + f_3 \mathbf{e}_3^a.$$

Friction in the UJOINT connection is the superposition of four HINGE-like frictional effects due to rotations about the two cross axes. Since the rotations about the local 1- and 3-directions are the only possible relative motions in the connection, the frictional effects (CSM1 and CSM3) are formally written in terms of moments generated by tangential tractions and moments generated by contact forces. In the following equations subscript 1 refers to frictional effects about the local 1-direction, and subscript 3 refers to frictional effects about the local 3-direction. The frictional effects are written as follows:

$$\Phi_1 = \mathrm{P}_1(\mathbf{f}) - \mu \mathrm{M}_{\mathrm{N}_1} \leq 0\,, \quad \text{and}$$
$$\Phi_3 = \mathrm{P}_3(\mathbf{f}) - \mu \mathrm{M}_{\mathrm{N}_3} \leq 0,$$

where the potentials $\mathrm{P}_1(\mathbf{f})$ and $\mathrm{P}_3(\mathbf{f})$ represent the moment magnitudes of the frictional tangential tractions in the connector in directions tangent to the cylindrical surface on which contact occurs, $\mathrm{M}_{\mathrm{N}_1}$ and $\mathrm{M}_{\mathrm{N}_3}$ are the friction-producing normal moments on the same cylindrical surface, and μ is the friction coefficient. Frictional stick occurs in a particular direction if $\Phi_1 < 0$ or $\Phi_3 < 0$; and sliding occurs if $\Phi_1 = 0$ or $\Phi_3 = 0$, in which case the friction moments are $\mu \mathrm{M}_{\mathrm{N}_1}$ and $\mu \mathrm{M}_{\mathrm{N}_3}$.

The normal moments $\mathrm{M}_{\mathrm{N}_1}$ and $\mathrm{M}_{\mathrm{N}_3}$ are the sums of magnitude measures of force-producing connector moments, $\mathrm{M}_{\mathrm{C}_1} = g_1(\mathbf{f})$ and $\mathrm{M}_{\mathrm{C}_3} = g_3(\mathbf{f})$, and self-equilibrated internal contact moments (such as from a press-fit assembly), $\mathrm{M}_{\mathrm{C}_1}^{\mathrm{int}}$ and $\mathrm{M}_{\mathrm{C}_3}^{\mathrm{int}}$, respectively:

$$\mathrm{M}_{\mathrm{N}_1} = 2\left|\mathrm{M}_{\mathrm{C}_1}\right| + \left|\mathrm{M}_{\mathrm{C}_1}^{\mathrm{int}}\right| = 2\left|g_1(\mathbf{f})\right| + \left|\mathrm{M}_{\mathrm{C}_1}^{\mathrm{int}}\right|, \quad \text{and}$$
$$\mathrm{M}_{\mathrm{N}_3} = 2\left|\mathrm{M}_{\mathrm{C}_3}\right| + \left|\mathrm{M}_{\mathrm{C}_3}^{\mathrm{int}}\right| = 2\left|g_3(\mathbf{f})\right| + \left|\mathrm{M}_{\mathrm{C}_3}^{\mathrm{int}}\right|.$$

The factor of two in the above equations comes from the fact that there are two hinges on each cross direction.

The moment magnitudes $\mathrm{M}_{\mathrm{C}_1}$ and $\mathrm{M}_{\mathrm{C}_3}$ are defined by summing the following contributions:

- moment from axial forces, $F_{axial\,1}^{hinge} R_a$ and $F_{axial\,3}^{hinge} R_a$, where $F_{axial\,1}^{hinge} = \alpha_{axial} F_{axial1}$, $F_{axial\,3}^{hinge} = \alpha_{axial} F_{axial3}$, and R_a is an average effective friction arm associated with the constraint force in the axial direction in each of the pins (if R_a is 0.0, $F_{axial\,1}^{hinge}$ and $F_{axial\,3}^{hinge}$ are ignored); and
- moment from normal forces, $F_{n1} R_p$ and $F_{n3} R_p$, where F_{n1} and F_{n3} are themselves sums of the following contributions:
  - transverse force contributions, $F_{total\ 1}^{hinge}$ (the magnitude of the total transverse force in the two hinges along the $\mathbf{e}_1^a$-direction) and $F_{total\ 3}^{hinge}$ (the magnitude of the total transverse force in the two hinges along the $\mathbf{e}_3^b$-direction):

$$F_{total\ 1}^{hinge} = \sqrt{\left(F_{bend}^{hinge}\right)^2 + \left(F_{twist}^{hinge}\right)^2 + \left(F_{transv1}^{hinge}\right)^2}\,, \quad \text{and}$$

$$F_{total\ 3}^{hinge} = \sqrt{\left(F_{bend}^{hinge}\right)^2 + \left(F_{twist}^{hinge}\right)^2 + \left(F_{transv3}^{hinge}\right)^2},$$

where $F_{bend}^{hinge} = \frac{F_{bend}}{4}$, F_{twist}^{hinge} is defined above, $F_{transv1}^{hinge} = \beta_{axial} F_{axial3}$, and $F_{transv3}^{hinge} = \beta_{axial} F_{axial1}$; and

- force contributions from "bending," F_{total}^{bend}, obtained by scaling the total bending moment, M_{total}^{hinge} (the magnitude of the total bending moment on each of the four hinges), by a length factor, as follows:

$$M_{total}^{hinge} = \sqrt{M_{bend}^{hinge2} + M_{twist}^{hinge2}},$$

$$F_{total}^{bend} = 2\frac{M_{total}^{hinge}}{L_s},$$

where $M_{bend}^{hinge} = \frac{1}{8} F_{bend} L_a$, M_{twist}^{hinge} is defined above, and L_s represents a characteristic overlapping length between the pins and their sleeves. If L_s is 0.0, M_{total}^{hinge} is ignored.

Thus,

$$\begin{aligned}
\mathrm{M}_{\mathrm{C}_1} = g_1(\mathbf{f}) &= F_{axial\,1}^{hinge} R_a + F_{n1} R_p \\
&= F_{axial\,1}^{hinge} R_a + \left(F_{total\ 1}^{hinge} + F_{total}^{bend}\right) R_p \\
&= \alpha_{axial} F_{axial1} R_a + R_p \sqrt{\left(\frac{F_{bend}}{4}\right)^2 + \left(F_{twist}^{hinge}\right)^2 + \left(\beta_{axial} F_{axial3}\right)^2} \\
&\quad + \frac{2R_p}{L_s} \sqrt{\left(\frac{1}{8} F_{bend} L_a\right)^2 + \left(M_{twist}^{hinge}\right)^2}, \quad \text{and} \\
\mathrm{M}_{\mathrm{C}_3} = g_3(\mathbf{f}) &= F_{axial\,3}^{hinge} R_a + F_{n3} R_p \\
&= F_{axial\,3}^{hinge} R_a + \left(F_{total\ 3}^{hinge} + F_{total}^{bend}\right) R_p \\
&= \alpha_{axial} F_{axial3} R_a + R_p \sqrt{\left(\frac{F_{bend}}{4}\right)^2 + \left(F_{twist}^{hinge}\right)^2 + \left(\beta_{axial} F_{axial1}\right)^2} \\
&\quad + \frac{2R_p}{L_s} \sqrt{\left(\frac{1}{8} F_{bend} L_a\right)^2 + \left(M_{twist}^{hinge}\right)^2}.
\end{aligned}$$

The moment magnitudes of the frictional tangential tractions are $\mathrm{P}_1(\mathbf{f}) = |m_1|$ and $\mathrm{P}_3(\mathbf{f}) = |m_3|$.

Summary

UJOINT	
Basic, assembled, or complex:	Assembled
Kinematic constraints:	JOIN + UNIVERSAL
Constraint force and moment output:	f_1, f_2, f_3, m_2
Available components:	ur_1, ur_3

UJOINT	
Kinetic force and moment output:	m_1, m_3
Orientation at *a*:	Required
Orientation at *b*:	Optional
Connector stops:	$\theta_1^{min} \le \gamma \le \theta_1^{max}$
	$\theta_3^{min} \le \gamma \le \theta_3^{max}$
Constitutive reference lengths:	$\theta_1^{ref}, \theta_3^{ref}$
Predefined friction parameters:	Required: R_p, L_a; optional: R_a, L_s, $\mathrm{M}_{\mathrm{C}_1}^{\mathrm{int}}$, $\mathrm{M}_{\mathrm{C}_3}^{\mathrm{int}}$
Contact moments for predefined friction:	$\mathrm{M}_{\mathrm{C}_1}$, $\mathrm{M}_{\mathrm{C}_3}$

UNIVERSAL

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type UNIVERSAL provides a connection between two nodes where the rotations are fixed about one local direction and free about two others.
Connection type UNIVERSAL provides the rotational part of a UJOINT connection. Connection type UNIVERSAL cannot be used in two-dimensional or axisymmetric analysis.

Description

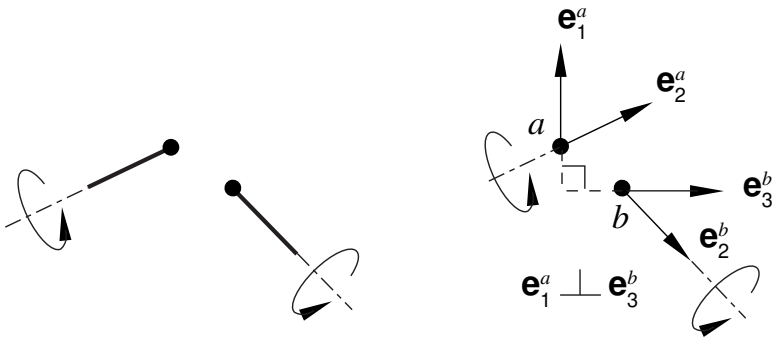


Figure 1: Connection type UNIVERSAL.

A UNIVERSAL connection constrains the rotation about the shaft directions at two nodes. The shaft directions at nodes a and b are $\mathbf{e}_2^a$ and $\mathbf{e}_2^b$, respectively. A UNIVERSAL connection requires that local direction $\mathbf{e}_1^a$ be perpendicular to $\mathbf{e}_3^b$. This single constraint is written

$$\mathbf{e}_1^a \cdot \mathbf{e}_3^b = 0 .$$

This constraint is equivalent to constraining the second Cardan angle to be zero in a Cardan angle parameterization of the local directions at node b relative to those at node a. If the initial orientation directions at node b do not satisfy the above constraint condition, the universal constraint will hold the second Cardan angle fixed at its initial value.

The constraint moment imposed by the UNIVERSAL connection is

$$\overline{\mathbf{m}} = m_2 \left(\cos\alpha \mathbf{e}_2^a + \sin\alpha \mathbf{e}_3^a \right) .$$

A UNIVERSAL connection allows two free rotational components of relative motion between two nodes. The first and third Cardan angles that position local directions at node b relative to those at node a are

$$\alpha = -\tan^{-1}\left(\frac{\mathbf{e}_2^a \cdot \mathbf{e}_3^b}{\mathbf{e}_3^a \cdot \mathbf{e}_3^b} \right) \quad \text{and} \quad \gamma = -\tan^{-1}\left(\frac{\mathbf{e}_1^a \cdot \mathbf{e}_2^b}{\mathbf{e}_1^a \cdot \mathbf{e}_1^b} \right) .$$

The two available components of relative motion for the UNIVERSAL connection, ur_1 and ur_3, are the changes in the two unconstrained Cardan angles when the second Cardan angle is fixed. Therefore,

$$ur_1 = \alpha - \alpha_0 \quad \text{and} \quad ur_3 = \gamma - \gamma_0 \,,$$

where α_0 and γ_0 are the initial Cardan angles. The connector constitutive rotations are

$$ur_1^{mat} = \alpha - \theta_1^{ref} \quad \text{and} \quad ur_3^{mat} = \gamma - \theta_3^{ref} \,.$$

The kinetic moment in the UNIVERSAL connection is

$$\mathbf{m}_{universal} = m_1 \mathbf{e}_1^a + m_3 \mathbf{e}_3^b \,.$$

Friction

When used by itself, there is no predefined Coulomb-like friction in the UNIVERSAL connection. However, when the UNIVERSAL connection is used in combination with the JOIN connection type, the predefined friction is the same as the UJOINT connection.

Summary

UNIVERSAL	
Basic, assembled, or complex:	Basic
Kinematic constraints:	$\mathbf{e}_1^a \cdot \mathbf{e}_3^b = 0$
Constraint moment output:	m_2
Available components:	ur_1, ur_3
Kinetic moment output:	m_1, m_3
Orientation at *a*:	Required
Orientation at *b*:	Optional
Connector stops:	$\theta_1^{min} \le \alpha \le \theta_1^{max}$,
	$\theta_3^{min} \le \gamma \le \theta_3^{max}$
Constitutive reference angles:	$\theta_1^{ref}, \theta_3^{ref}$
Predefined friction parameters:	None
Contact force for predefined friction:	None

WELD

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Connector Elements*
- *Connector Element Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR SECTION*

Overview

Connection type WELD provides a fully bonded connection between two nodes.

Description

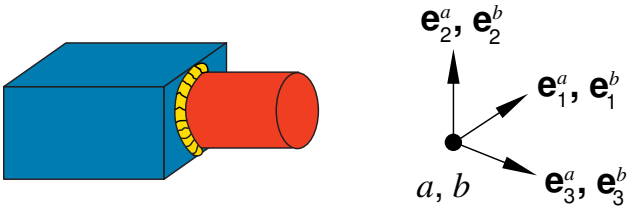


Figure 1: Connection type WELD.

Connection type WELD imposes kinematic constraints and uses local orientation definitions equivalent to combining connection types JOIN and ALIGN.

Summary

WELD	
Basic, assembled, or complex:	Assembled
Kinematic constraints:	JOIN + ALIGN
Constraint force and moment output:	$f_1, f_2, f_3, m_1, m_2, m_3$
Available components:	None
Kinetic force and moment output:	None
Orientation at *a*:	Optional
Orientation at *b*:	Optional
Connector stops:	None
Constitutive reference lengths and angles:	None
Predefined friction parameters:	None
Contact force for predefined friction:	None

Connector Element Behavior

In this section:

- *Connector Behavior*
- *Connector Elastic Behavior*
- *Connector Damping Behavior*
- *Connector Functions for Coupled Behavior*
- *Connector Friction Behavior*
- *Connector Plastic Behavior*
- *Connector Damage Behavior*
- *Connector Stops and Locks*
- *Connector Failure Behavior*
- *Connector Uniaxial Behavior*

Connector Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Connectors*
- *Connection-Type Library*
- *Connection Types*
- *Connector Elastic Behavior*
- *Connector Damping Behavior*
- *Connector Functions for Coupled Behavior*
- *Connector Friction Behavior*
- *Connector Plastic Behavior*
- *Connector Damage Behavior*
- *Connector Stops and Locks*
- *Connector Failure Behavior*
- *Connector Uniaxial Behavior*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR CONSTITUTIVE REFERENCE*
- **CONNECTOR SECTION*
- *Creating connector sections*
- *Defining a reference length*
- *Defining time integration*

Overview

Connector behavior:

- can be defined for connection types with available components of relative motion;
- can incorporate simple spring, dashpot, and node-to-node contact as particular applications;
- may include linear or nonlinear force versus displacement and force versus velocity behavior for the unconstrained relative motion components;
- can include uncoupled or coupled behavior specifications;
- can allow frictional force in an unconstrained component of relative motion to be generated by any force or moment in the connection;
- can allow for plasticity definitions for individual components or coupled plasticity definitions using user-defined yield functions;
- can be used to specify sophisticated damage mechanisms with various damage evolution laws;
- can provide user-defined locking criteria to lock in the current position all relative motion in the connector element or a single unconstrained component of relative motion;
- can be used to specify failure of the connector element; and
- can be used to specify complex uniaxial models by specifying the loading and unloading behavior in an available component of relative motion.

Assigning a Connector Behavior to a Connector Element

You can assign the name of a connector behavior to particular connector elements.

Input File Usage: Use the following options to define the connector behavior:

**CONNECTOR SECTION*, ELSET=*name*, BEHAVIOR=*behavior name*
**CONNECTOR BEHAVIOR*, NAME=*behavior name*

Abaqus/CAE Usage: Interaction module:
Connector->Section->Create: **Name:** *connector section name*:
Behavior Options, **Add**
Connector->Assignment->Create: select wires: **Section**: *connector section name*

Connector Behavior Models

Connector behaviors allow for modeling of the following types of effects:

- spring-like elastic behavior;
- rigid-like elastic behavior;
- dashpot-like (damping) behavior;
- friction;
- plasticity;
- damage;
- stops;
- locks;
- failure; and
- uniaxial behavior.

Kinetic behavior can be specified only in available components of relative motion. The list of available components of relative motion for each connector type is given in *Connection-Type Library*. A connector behavior can be specified in any of the following ways:

- uncoupled: the behavior is specified separately in individual available components of relative motion;
- coupled: all or several of the available components of relative motion are used simultaneously in a coupled manner to define the behavior; or
- combined: a combination of both uncoupled and coupled definitions are used simultaneously.

A conceptual model illustrating how connector behaviors interact with each other is shown in *Figure 1*. Most behaviors (elasticity, damping, stops, locks, friction) act in parallel. Plasticity models are always defined in conjunction with spring-like or rigid-like elasticity definitions. Degradation due to damage can be specified either for the elastic-plastic or rigid-plastic response alone or for the entire kinetic response in the connector. The failure behavior will apply to the entire connector response.

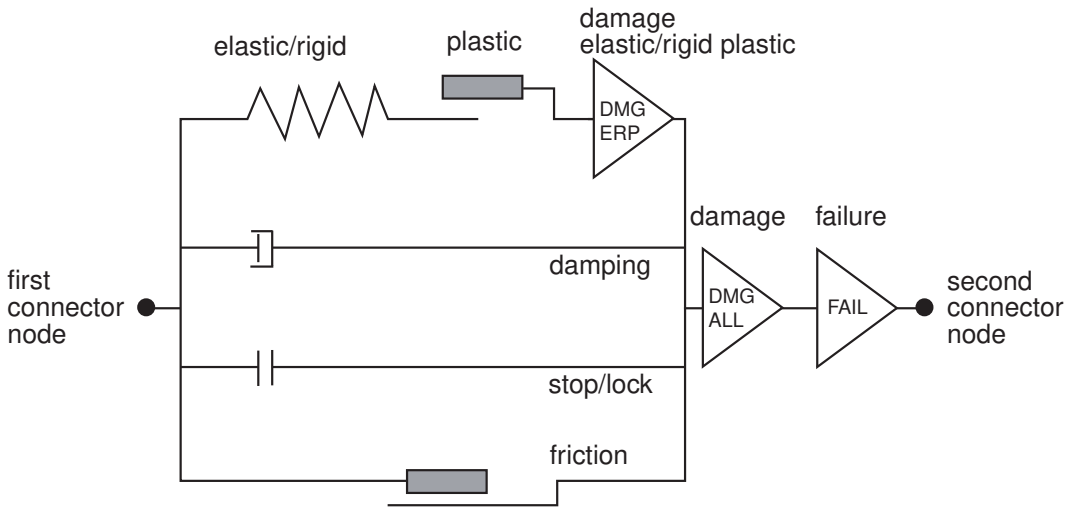


Figure 1: Conceptual illustration of connector behaviors.

Multiple definitions for the same behavior type are permitted. For example, if connector elasticity (or damping) is defined several times in an uncoupled fashion for the same available component of relative motion, in a coupled fashion, or in both fashions, the spring-like (or dashpot-like) responses are added together. Multiple definitions of friction, plasticity, and damage behaviors are permitted as long as the rules outlined in the corresponding behavior sections are followed. Multiple uncoupled stop and lock definitions for the same component are permitted, but only one will be enforced at a time.

Defining Coupled and Uncoupled Connector Behavior

In many cases connector behavior is specified in an uncoupled manner in individual available components of relative motion. Coupled behavior can be defined for all or some of the available components of relative motion in a connector.

For coupled plasticity, damage, and, in certain situations, friction behavior, additional functions describing the nature of the coupling effects must be defined (see *Connector Functions for Coupled Behavior*). These functions do not define a behavior by themselves but are used as tools for building a desired behavior. For example, these functions may be used to define:

- sophisticated yield functions in the connector force space for coupled plasticity behavior;
- friction-generating contact forces for friction behavior; or
- force or relative motion magnitude measures needed for damage behavior specifications.

Input File Usage: Use the following input to define uncoupled behavior:

**CONNECTOR BEHAVIOR OPTION*, COMPONENT=*n*

Use the following input to define coupled behavior:

**CONNECTOR BEHAVIOR OPTION*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->*connector behavior***: **Coupling: Uncoupled** or **Coupled**

Defining Nonlinear Connector Behavior Properties to Depend on Relative Positions or Constitutive Displacements/Rotations

In all nonlinear uncoupled connector kinetic behaviors the independent variable is the connector available component in the direction for which the response is defined. When modeling the following connector behaviors, the properties can also depend on relative positions or constitutive displacements/rotations in several component directions:

- connector elasticity,
- connector damping,
- connector derived components, and
- connector friction.

When modeling connector uniaxial behavior, the properties can also depend on constitutive displacements/rotations in several component directions; see *Connector Uniaxial Behavior* for more information.

Input File Usage: Use the following option to specify that the connector behavior properties are dependent on components of relative position included in the behavior definition:

**CONNECTOR BEHAVIOR OPTION*,
INDEPENDENT COMPONENTS=POSITION (default)

Use the following option to specify that the connector behavior properties are dependent on components of constitutive relative displacements or rotations included in the behavior definition:

**CONNECTOR BEHAVIOR OPTION*,
INDEPENDENT COMPONENTS=CONSTITUTIVE MOTION

In either case the first data line identifies the independent component numbers to be used in determining the dependencies, and the additional data for the connector behavior definition begin on the second data line.

Abaqus/CAE Usage: For elasticity or damping behavior, use the following input to specify that connector behavior properties are dependent on relative position or constitutive relative displacements/rotations:

Interaction module: connector section editor: **Add->Elasticity** or **Damping**: **Coupling: Coupled on position** or **Coupled on motion**, select components and enter data

For connector derived components, use the following input to specify that connector behavior properties are dependent on relative position or constitutive relative displacements/rotations:

Interaction module: connector section editor:
Add->Friction, **Plasticity**, or **Damage**: **Force Potential**, **Initiation Potential**, or **Evolution Potential**
Specify derived component, **Use local directions**: **Independent position components** or **Independent constitutive motion components**, select components and enter data

For friction behavior specifying internal contact forces, use the following input to specify that connector behavior properties are dependent on relative position or constitutive relative displacements/rotations:

Interaction module: connector section editor: **Add->Friction**: **Friction model**: **User-defined**, **Contact Force**, **Use independent components: Position** or **Motion**, select components and enter data

Defining Reference Lengths and Angles for Constitutive Response

In many connector behavior definitions, material-like behavior has a reference position where the force or moment is zero, which is different from the initial position. This is the case, for example, in a spring that has nonzero force or moment in the initial configuration. In these situations the most convenient way to define the connector behavior is relative to the nominal or reference geometry where the forces or moments vanish.

You can define the translational or angular positions at which constitutive forces and moments are zero by specifying up to six reference values (one per component of relative motion): three lengths and three angles (in degrees). The reference lengths and angles affect only spring-like elastic connector behavior and, if the friction-generating contact force (moment) is a function of the relative displacement (rotation), connector friction behavior. By default, the reference lengths and angles are the length and angle values determined from the initial geometry. See *Connection Types* for the meaning of the reference lengths and angles for each connection type. Abaqus will not account for the energy stored as a result of precompressed or preextended behavior.

Input File Usage: **CONNECTOR CONSTITUTIVE REFERENCE*
length 1, *length 2*, *length 3*, *angle 1*, *angle 2*, *angle 3*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Reference Length**: **Length associated with *CORM***

Defining Precompressed or Preextended Linear Elastic Behavior

In many cases connectors are precompressed or preextended when installed in assemblies. In such cases the connector force is nonzero in the initial configuration. While nonlinear elasticity could be used to define nonzero force in the initial configuration, it is often more convenient to specify a (linear) spring stiffness plus a reference length or angle at which the force or moment is zero. For example, linear uncoupled elastic behavior defined with the connection type AXIAL would have force given by the equation

$$F_1 = D_{11} u_1^{mat},$$

where $u_1^{mat} = l - l_1^{ref}$. l is the current length of the AXIAL connection, and l_1^{ref} is the user-defined constitutive reference length. The connector constitutive displacement quantities, u_j^{mat}, are defined for different connection types as described in *Connection Types*.

Example

An input file template for a connector model of the shock absorber in *Figure 2* is presented in *About Connectors*. A reference angle of 22.5° is defined for the nonlinear torsional spring as the fourth data item (corresponding to the connector's fourth component of relative motion) in the connector constitutive reference:

```
*CONNECTOR BEHAVIOR, NAME=sbehavior
...
*CONNECTOR CONSTITUTIVE REFERENCE
 , , , 22.5
```

The effect of this reference angle is that the nonlinear torsional spring has a zero moment at an angle of 22.5°.

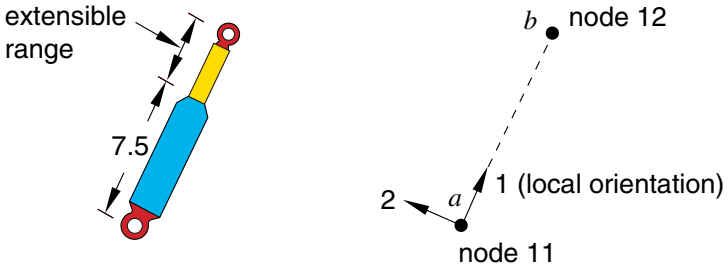


Figure 2: Simplified connector model of a shock absorber.

Defining the Time Integration Method for Constitutive Response in Abaqus/Explicit

In Abaqus/Explicit kinematic constraints, stops, locks, and actuated motion in connector elements are treated with implicit time integration. By default, connector constitutive behavior (for example, elasticity, damping, and friction) is also integrated implicitly. The advantage of implicit time integration is that elements with these behaviors do not affect the stability or time incrementation of the analysis in any way.

When "soft" springs are modeled with connectors, a more traditional explicit time integration for the constitutive response can be used. This explicit time integration may lead to a small improvement in computational performance. However, explicit integration of relatively stiff springs will reduce the global time increment size, since such connector elements are included in the stable time increment size calculation.

Input File Usage: Use the following option to specify implicit integration of the constitutive response:

**CONNECTOR BEHAVIOR*, INTEGRATION=IMPLICIT

Use the following option to specify explicit integration of the constitutive response:

**CONNECTOR BEHAVIOR*, INTEGRATION=EXPLICIT

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Integration**: **Integration: Implicit** or **Explicit**

Defining Connector Behavior in Linear Perturbation Procedures

In linear perturbation procedures (see *General and Perturbation Procedures*) the connector element kinematics are linearized about the base state. Hence, linearized versions of kinematic constraints are applied, and the connector behavior is linearized about the state at the end of the previous general analysis step.

Using Several Connectors in Series or in Parallel

Connector element behaviors allow for proper modeling of most physical connection behaviors within a single connector element. However, in rare circumstances more complex connection behaviors may require multiple connector elements to be used in parallel or in series. You place connector elements in parallel by defining two or more connector elements between the same nodes. You place connectors in series by specifying additional nodes (most often in the same location as the nodes of interest) and then stringing connector elements between these nodes.

For example, assume that you would like to define a connector stop that exhibits elastic-plastic behavior upon contact. Since this is not permitted within the context of one connector behavior definition, you can circumvent the limitation by using two connector elements in series. This concept is illustrated in *Figure 3*. The first connector

defines the stop, and the second defines the elastic-plastic behavior. Since both elements are subject to the same load (because they are in series), the desired behavior is obtained.

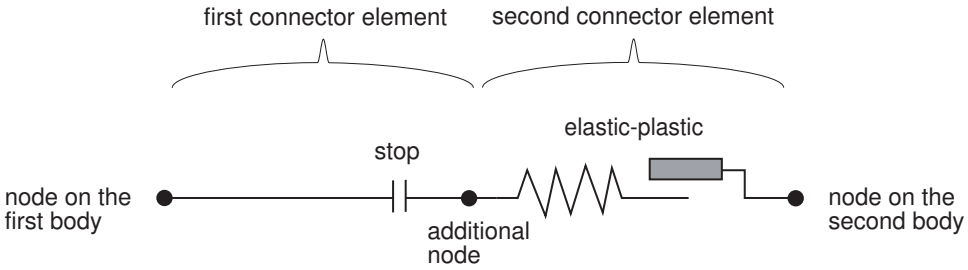


Figure 3: Conceptual illustration of two connector elements/behaviors in series.

Connectors in parallel can be used as well to model complex kinetic behavior. For example, assume that you need to define an elastic-viscous connector with spring-like and dashpot-like behaviors in parallel (for example, the strut in an automotive suspension). Assume that damage can occur only in the dashpot once it is stretched/compressed beyond specified limits. Since this is not permitted within the context of one connector behavior definition, you can circumvent the limitation by using two connector elements in parallel. This concept is illustrated in *Figure 4*.

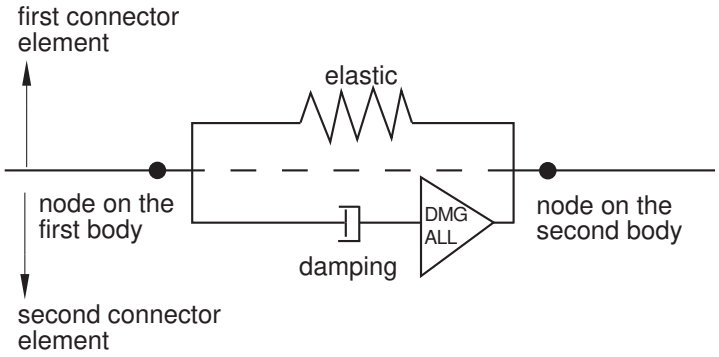


Figure 4: Conceptual illustration of two connector elements/behaviors in parallel.

The first connector defines the elastic behavior, and the second defines the dashpot behavior. Since the two connector elements are in parallel, they undergo the same motion (stretching/compression). A motion-based damage behavior (see *Connector Damage Behavior*) can be used to degrade the entire behavior in the second element. Thus, only the dashpot behavior will eventually degrade.

Defining Connector Behavior Using Tabular Data

Tabular data are often used to define connector behaviors, such as nonlinear elasticity, isotropic hardening, etc. As shown in *Figure 5*, the data points make up a nonlinear curve in the constitutive space.

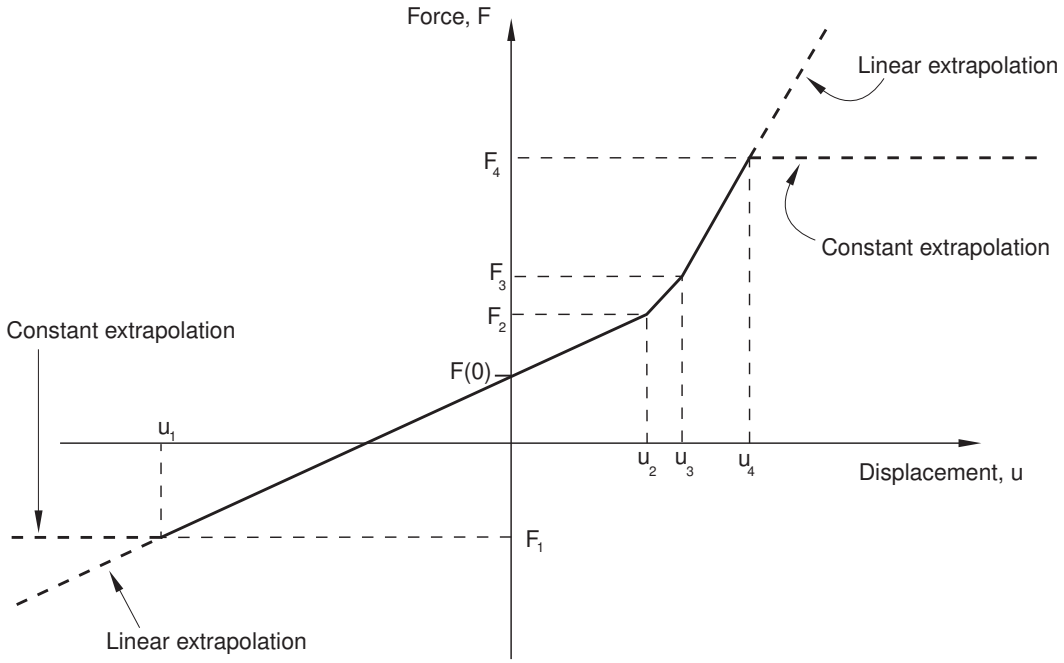


Figure 5: Nonlinear connector behaviors defined as tabular data.

The options to define table lookups are described below.

Extrapolation Options

By default, the dependent variables are extrapolated as a constant (with a value corresponding to the endpoints of the curve) outside the specified range of the independent variables. This choice may cause a zero stiffness response, which may lead to convergence problems. You can specify linear extrapolation to extrapolate the dependent variables outside the specified range of the independent variables assuming that the slope given by the end points of the curve remains constant. The extrapolation behavior is illustrated in *Figure 5*.

You define the extrapolation choice globally for all connector behaviors but can redefine the extrapolation choice for the following connector behaviors individually:

- connector elasticity;
- connector plasticity (connector hardening);
- connector damping;
- derived components for connector elements;
- connector friction;
- connector damage (connector damage initiation and evolution);
- connector locks; and
- connector uniaxial behavior.

Tabular data for connector stop and lock behavior options are not supported in Abaqus/CAE.

Specifying Constant Extrapolation for All Connector Behaviors

You can specify constant extrapolation for tabular data for all connector behaviors.

Input File Usage: **CONNECTOR BEHAVIOR*, EXTRAPOLATION=CONSTANT (default)

Abaqus/CAE Usage: Interaction module: connector section editor: **Table Options** tabbed page: **Extrapolation**: **Constant**

Specifying Linear Extrapolation for All Connector Behaviors

You can specify linear extrapolation for tabular data for all connector behaviors.

Input File Usage: **CONNECTOR BEHAVIOR*, EXTRAPOLATION=LINEAR

Abaqus/CAE Usage: Interaction module: connector section editor: **Table Options** tabbed page: **Extrapolation**: **Linear**

Redefining the Extrapolation Choice for Individual Connector Behaviors

You can redefine the extrapolation choice for individual connector behaviors.

Input File Usage: Use either of the following options:

**CONNECTOR BEHAVIOR OPTION*, EXTRAPOLATION=CONSTANT

**CONNECTOR BEHAVIOR OPTION*, EXTRAPOLATION=LINEAR

For example, use the following options to use constant extrapolation for all connector behaviors except for connector elasticity:

**CONNECTOR BEHAVIOR*, EXTRAPOLATION=CONSTANT

**CONNECTOR ELASTICITY*, EXTRAPOLATION=LINEAR

Abaqus/CAE Usage: Use the following input for elasticity, damping, friction, plasticity, and damage behaviors:

Interaction module: connector section editor: **Behavior Options** tabbed page: **Table Options** button: **Extrapolation**: toggle off **Use behavior settings** and choose **Constant** or **Linear**

Use the following input for connector derived components:

Interaction module: derived component editor: **Add**: **Table Options** button: **Extrapolation**: toggle off **Use behavior settings** and choose **Constant** or **Linear**

Regularization Options for Abaqus/Explicit

By default, Abaqus/Explicit regularizes the data into tables that are defined in terms of even intervals of the independent variables since table lookups are most economical if the interpolation is from even intervals of the independent variables. In some cases, where it is necessary to capture sharp changes in connector behavior accurately, you can use the user-defined tabular connector behavior data directly by turning regularization off. However, the table lookups will be more computationally expensive compared to using regular intervals. Therefore, the use of regularization is almost always recommended.

Abaqus/Explicit uses an error tolerance to regularize the input data. The number of intervals in the range of each independent variable is chosen such that the error between the piecewise linear regularized data and each of your defined points is less than the tolerance times the range of the dependent variable. The default tolerance is 0.03.

In some cases where the dependent quantities are defined at uneven intervals of the independent variables and the range of the independent variable is large compared to the smallest interval, Abaqus/Explicit may fail to obtain an accurate regularization of your data in a reasonable number of intervals. In this case Abaqus/Explicit stops after all data are processed and issues an error message that you must redefine the behavior data. See *Material Data Definition* for a more detailed discussion of data regularization.

You define the choice of regularization and regularization tolerance globally for all connector behaviors but can redefine the choice of regularization and regularization tolerance for the following connector behaviors individually:

- connector elasticity;
- connector plasticity (connector hardening)
- connector damping;
- derived components for connector elements;
- connector friction;
- connector damage (connector damage initiation and evolution);
- connector locks; and
- connector uniaxial behavior.

Tabular data for connector stop and lock behavior options are not supported in Abaqus/CAE.

Specifying the Regularization of User-Defined Tabular Data for All Connector Behaviors

You can specify regularization of tabular data and a regularization tolerance to be used globally for all connector behaviors.

Input File Usage: **CONNECTOR BEHAVIOR*, REGULARIZE=ON (default), RTOL=*tolerance*

Abaqus/CAE Usage: Interaction module: connector section editor: **Table Options** tabbed page: **Regularization**: toggle on **Regularize data (Explicit only)**, **Specify**: *tolerance*

Specifying the Use of User-Defined Tabular Data without Regularization for All Connector Behaviors

You can specify the use of user-defined tabular data directly by turning regularization off for all connector behaviors.

Input File Usage: **CONNECTOR BEHAVIOR*, REGULARIZE=OFF

Abaqus/CAE Usage: Interaction module: connector section editor: **Table Options** tabbed page: **Regularization**: toggle off **Regularize data (Explicit only)**

Redefining the Regularization Options for Individual Connector Behaviors

You can redefine the choice of regularization and regularization tolerance for individual connector behaviors.

Input File Usage: Use either of the following options:

**CONNECTOR BEHAVIOR OPTION*, REGULARIZE=ON, RTOL=*tolerance*

**CONNECTOR BEHAVIOR OPTION*, REGULARIZE=OFF

For example, use the following options to regularize the user-defined data for all connector behaviors except for connector elasticity:

**CONNECTOR BEHAVIOR*, REGULARIZE=ON, RTOL=0.05

**CONNECTOR ELASTICITY*, REGULARIZE=OFF

Abaqus/CAE Usage: Use the following input for elasticity, damping, friction, plasticity, and damage behaviors:

Interaction module: connector section editor: **Behavior Options** tabbed page: **Table Options** button: **Regularization**: toggle off **Use behavior settings**; toggle on **Regularize data (Explicit only)** and **Specify**: *tolerance*, or toggle off **Regularize data (Explicit only)**

Use the following input for connector derived components:

Interaction module: derived component editor: **Add**: **Table Options** button: **Regularization**: toggle off **Use behavior settings**; toggle on **Regularize data (Explicit only)** and **Specify**: *tolerance*, or toggle off **Regularize data (Explicit only)**

Evaluation of Rate-Dependent Data

Data for the tabulated isotropic hardening in connector plasticity (*Defining the Isotropic Hardening Component by Specifying Tabular Data*) and plastic motion–based damage initiation criterion (*Plastic Motion–Based Damage Initiation Criterion*) can be specified as dependent on the equivalent relative plastic motion rate. Loading/unloading data for the rate-dependent connector uniaxial behavior model can be specified as dependent on the rate of deformation.

Specifying Linear Intervals for Interpolation of Rate-Dependent Data

By default, both Abaqus/Standard and Abaqus/Explicit interpolate rate-dependent data using linear intervals of the relative motion rate.

Input File Usage: Use the following option to specify linear interpolation for isotropic hardening data:

**CONNECTOR HARDENING*, RATE INTERPOLATION=LINEAR

Use the following option to specify linear interpolation for damage initiation data:

**CONNECTOR DAMAGE INITIATION*, RATE INTERPOLATION=
LINEAR

Use both of the following options to specify linear interpolation for uniaxial behavior loading/unloading data:

**CONNECTOR UNIAXIAL BEHAVIOR*
**LOADING DATA*, RATE INTERPOLATION=LINEAR

Abaqus/Standard always interpolates rate-dependent data using linear intervals of the equivalent relative plastic motion rate.

Abaqus/CAE Usage: Use the following input for isotropic hardening data:

Interaction module: connector section editor: **Add->Plasticity**: **Isotropic Hardening**: **Definition**: **Tabular**, **Table Options** button: **Interpolation**: **Linear**

Use the following input for damage initiation data:

Interaction module: connector section editor: **Add->Damage**: **Initiation**: **Table Options** button: **Interpolation**: **Linear**

Connector uniaxial behavior cannot be defined in Abaqus/CAE.

Specifying Logarithmic Intervals for Interpolation of Rate-Dependent Data in Abaqus/Explicit

In Abaqus/Explicit you can specify that logarithmic intervals of the relative motion rate be used for the interpolation of rate-dependent data if the rate dependence of the data is measured at logarithmic intervals.

Input File Usage: Use the following option to specify linear interpolation for isotropic hardening data:

**CONNECTOR HARDENING*, RATE INTERPOLATION=LOGARITHMIC

Use the following option to specify linear interpolation for damage initiation data:

**CONNECTOR DAMAGE INITIATION*, RATE INTERPOLATION=LOGARITHMIC

Use both of the following options to specify linear interpolation for uniaxial behavior loading/unloading data:

**CONNECTOR UNIAXIAL BEHAVIOR*
**LOADING DATA*, RATE INTERPOLATION=LOGARITHMIC

Abaqus/CAE Usage: Use the following input for isotropic hardening data:

Interaction module: connector section editor: **Add->Plasticity**: **Isotropic Hardening**: **Definition**: **Tabular**, **Table Options** button: **Interpolation**: **Logarithmic**

Use the following input for damage initiation data:

Interaction module: connector section editor: **Add->Damage**: **Initiation**: **Table Options** button: **Interpolation**: **Logarithmic**

Connector uniaxial behavior cannot be defined in Abaqus/CAE.

Filtering the Equivalent Plastic Motion Rate in Abaqus/Explicit

Rate-sensitive connector constitutive behavior may introduce nonphysical high-frequency oscillations in an explicit dynamic analysis. To overcome this problem, Abaqus/Explicit uses a filtered equivalent plastic motion rate

$$\dot{\bar{u}}^{pl}\Big|_{t+\Delta t} = \omega\ \frac{\Delta \bar{u}^{pl}}{\Delta t} + (1-\omega)\ \dot{\bar{u}}^{pl}\Big|_{t}$$

for the evaluation of rate-dependent data. $\Delta\bar{u}^{pl}$ is the incremental change in equivalent plastic motion during the time increment Δt, and $\dot{\bar{u}}^{pl}\big|_t$ and $\dot{\bar{u}}^{pl}\big|_{t+\Delta t}$ are the plastic motion rates at the beginning and end of the increment, respectively. The factor ω ($0 < \omega \leq 1$) facilitates filtering high-frequency oscillations associated with rate-dependent connector behavior. You can specify the value of the rate filter factor, ω, directly. The default value is 0.9. A value of $\omega = 1$ provides no filtering and should be used with caution.

Input File Usage: Use either of the following options:

**CONNECTOR HARDENING*, RATE FILTER FACTOR=ω

**CONNECTOR DAMAGE INITIATION*, RATE FILTER FACTOR=ω

Abaqus/CAE Usage: Use the following input for isotropic hardening data:

Interaction module: connector section editor: **Add**->**Plasticity**: **Isotropic Hardening**: **Definition**: **Tabular**, **Table Options** button: **Filter factor**: **Specify**: ω

Use the following input for damage initiation data:

Interaction module: connector section editor: **Add**->**Damage**: **Initiation**: **Table Options** button: **Filter factor**: **Specify**: ω

Connector Elastic Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Connectors*
- *Connection Types*
- *Connector Behavior*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR ELASTICITY*
- *Defining elasticity*

Overview

Spring-like elastic connector behavior:

- can be defined in any connector with available components of relative motion;
- can be specified for each available component of relative motion independently, in which case the behavior can be linear or nonlinear;
- can be specified as dependent on relative positions or constitutive motions in several local directions; and
- can be specified for all available components of relative motion as coupled linear elastic behavior.

Alternatively, rigid-like behavior can be specified in any of the available components of relative motion using an automatically chosen stiff spring.

The directions in which the forces and moments act and the displacements and rotations are measured are determined by the local directions as described in *Connection Types* for each connection type.

Defining Linear Uncoupled Elastic Behavior

In the simplest case of linear uncoupled elasticity you define the spring stiffnesses for the selected components (i.e., D_{11} for component 1, D_{22} for component 2, etc.), which are used in the equation

$$F_i = D_{ii} u_i \quad \text{(no sum on } i\text{)},$$

where F_i is the force or moment in the i^{th} component of relative motion and u_i is the connector displacement or rotation in the i^{th} direction. The elastic stiffness can depend on frequency (in Abaqus/Standard), temperature, and field variables. See *Input Syntax Rules* for further information about defining data as functions of frequency, temperature, and field variables.

In most cases if frequency-dependent damping behavior is specified in an Abaqus/Standard analysis procedure, the data at zero frequency is used. The exceptions are direct-solution steady-state dynamics, subspace-based steady-state dynamics, and natural or complex eigenvalue extraction.

Input File Usage: Use the following options to define linear uncoupled elastic connector behavior:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR ELASTICITY*, COMPONENT=*component number*, DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Elasticity**: **Definition: Linear**, **Force/Moment:** *component or components*, **Coupling: Uncoupled**

Defining Linear Coupled Elastic Behavior

In the linear coupled case you define the spring stiffness matrix components, D_{ij}, which are used in the equation

$$F_i = \sum_j D_{ij} u_j,$$

where F_i is the force in the i^{th} component of relative motion, u_j is the motion of the j^{th} component, and D_{ij} is the coupling between the i^{th} and j^{th} components. The *D* matrix is assumed to be symmetric, so only the upper triangle of the matrix is specified. In connectors with kinematic constraints the entries that correspond to the constrained components of relative motion will be ignored. The elastic stiffness can depend on temperature and field variables. See *Input Syntax Rules* for further information about defining data as functions of temperature and field variables.

Input File Usage: Use the following options to define linear coupled elastic connector behavior:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR ELASTICITY*, DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Elasticity**: **Definition: Linear**, **Force/Moment:** *component or components*, **Coupling: Coupled**

Modeling Coupled Unsymmetric Linear Stiffness

By definition, linear elastic behavior should be defined by a symmetric spring stiffness matrix. However, Abaqus/Standard allows you to define an unsymmetric coupled spring stiffness matrix. The intended use case is to approximate fluid film bearings supporting a rotating structure in a rotordynamic analysis (see *Genta, 2005*, and *Distributed Loads*). Abaqus/Standard will not check the stability of an unsymmetric spring stiffness matrix; therefore, you must ensure that it is defined properly.

In the linear coupled case you define the spring stiffness matrix components, D_{ij}, which are used in the equation

$$F_i = \sum_j D_{ij} u_j,$$

where F_i is the force in the i^{th} component of relative motion, u_j is the motion of the j^{th} component, and D_{ij} is the coupling between the i^{th} and j^{th} components. The *D* matrix in this case is assumed to be unsymmetric, so the entire matrix is specified. The entries that correspond to the constrained components of relative motion are ignored. When the unsymmetric matrix storage and solution scheme are used, the stiffness can depend on frequency, temperature, and field variables. See *Input Syntax Rules* for further information about defining data as functions of frequency, temperature and field variables.

Input File Usage: Use the following options to define unsymmetric linear coupled stiffness connector behavior:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR ELASTICITY*, UNSYMM,
FREQUENCY DEPENDENCE=ON

Abaqus/CAE Usage: Unsymmetric linear coupled stiffness behavior is not supported in Abaqus/CAE.

Defining Nonlinear Elastic Behavior

For nonlinear elasticity you specify forces or moments as nonlinear functions of one or more available components of relative motion, $F_i(u_1, u_2, \ldots)$. These functions can also depend on temperature and field variables. See *Input Syntax Rules* for further information about defining data as functions of temperature and field variables.

Defining Nonlinear Elastic Behavior That Depends on One Component Direction

By default, each nonlinear force or moment function depends only on the displacement or rotation in the direction of the specified component of relative motion.

Input File Usage: Use the following options:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR ELASTICITY*, COMPONENT=*component number*, NONLINEAR, DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Elasticity**: **Definition: Nonlinear**, **Force/Moment:** *component or components*, **Coupling: Uncoupled**

Defining Nonlinear Elastic Behavior That Depends on Several Component Directions

Alternatively, the functions can depend on the relative positions or constitutive displacements/rotations in several component directions, as described in *Defining Nonlinear Connector Behavior Properties to Depend on Relative Positions or Constitutive Displacements/Rotations*. In this case the operator matrices are unsymmetric when $\partial F_i / \partial u_j \neq \partial F_j / \partial u_i$, for $i \neq j$, and unsymmetric matrix storage and solution may be needed in Abaqus/Standard to improve convergence.

Input File Usage: Use the following options to define nonlinear elastic connector behavior that depends on components of relative position:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR ELASTICITY*, COMPONENT=*component number*, NONLINEAR, INDEPENDENT COMPONENTS=POSITION, DEPENDENCIES=*n*

Use the following options to define nonlinear elastic connector behavior that depends on components of constitutive displacements or rotations:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR ELASTICITY*, COMPONENT=*component number*, NONLINEAR, INDEPENDENT COMPONENTS=CONSTITUTIVE MOTION, DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Elasticity**: **Definition: Nonlinear**, **Force/Moment:** *component or components*, **Coupling: Coupled on position** or **Coupled on motion**

Examples

The combined connector in *Figure 1* has two available components of relative motion: the relative displacement along the 1-direction (from the SLOT connection) and the rotation around the 1-direction (from the REVOLUTE connection)—see *SLOT* and *REVOLUTE*. Thus, the connector components of relative motion 1 and 4 can be used to specify connector behavior.

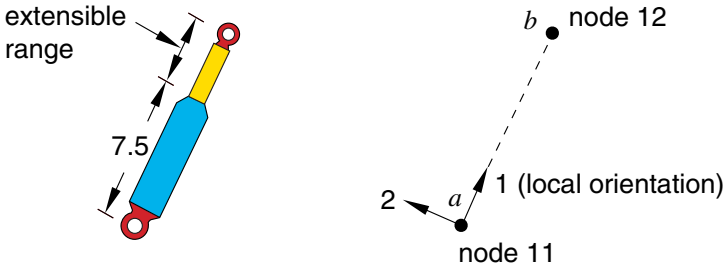


Figure 1: Simplified connector model of a shock absorber.

To define a nonlinear torsional spring to resist the relative rotation between the top and the bottom connection point around the local 1-direction, use the following input:

```
*CONNECTOR SECTION, ELSET=shock, BEHAVIOR=sbehavior
slot, revolute
ori,
*CONNECTOR BEHAVIOR, NAME=sbehavior
*CONNECTOR ELASTICITY, COMPONENT=4, NONLINEAR
-900., -0.7
   0.,  0.0
1250.,  0.7
```

Although no elastic coupling is assumed to occur between the two available components of relative motion, you could replace the nonlinear moment versus rotation data with coupled linear elastic behavior to define the rotational stiffness around the shock's axis coupled to the axial displacement.

In another application this same connector may have coupled linear elastic behavior, in the sense that relative rotation and sliding affect each other through a linear coupling. To define a translational stiffness of 2000.0 units, the D_{11} constant (the 1st entry of a symmetric matrix) is entered in the connector elasticity definition. To define a torsional stiffness of 1000.0 units, the D_{44} constant (the 10th entry of a symmetric matrix) is entered; and to define a coupling stiffness of 50.0 units between the available rotation and displacement, the D_{14} constant (the 7th entry) is entered.

```
*CONNECTOR ELASTICITY
2000.0, , , , , , 50.0,
0.0, 1000.0, , , , , ,
, , , ,
```

Defining Rigid Connector Behavior

Rigid-like elastic connector behavior can be used to make an otherwise available component of relative motion rigid. Consider a CARTESIAN connector that has no intrinsic kinematic constraints. If rigid behavior is specified in the local 2- and 3-directions, the connector will behave in a similar fashion to a SLOT connector.

This technique of using connectors with available components of relative motion for which rigid behavior is specified instead of connectors with intrinsically kinematic constraints is particularly useful when you need to:

- customize the constrained components in a connector with available components of relative motion; for example, you can constrain the local 1- and 2-directions in a CARTESIAN connector to define a SLOT-like connector in the 3-direction;

- define rigid plastic behavior (see *Connector Plastic Behavior*); or
- define rigid damage behavior (see *Connector Damage Behavior*).

For example, if you use a SLOT connector, plasticity and damage behavior cannot be specified in the intrinsically constrained 2- and 3-directions. To resolve the issue, you can use a CARTESIAN connector with rigid behavior in components 2 and 3 as discussed above and then define rigid plasticity (and/or damage) in these components. See the examples in *Connector Plastic Behavior* for illustrations.

In Abaqus/Standard an overconstraint may occur if a rigid component is defined in the same local direction as an active connector stop, connector lock, or specified connector motion.

Input File Usage: Use the following option to define rigid connector behavior for a specified component of relative motion:

**CONNECTOR ELASTICITY*, RIGID, COMPONENT=*n*

Use the following option to define rigid connector behavior for multiple specified components of relative motion:

**CONNECTOR ELASTICITY*, RIGID
data line listing components to be made rigid

Use the following option to define rigid connector behavior for all available components of relative motion:

**CONNECTOR ELASTICITY*, RIGID
(no data lines)

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Elasticity**: **Definition: Rigid**, **Components:** *component or components*

Enforcing Rigid-Like Elastic Behavior

Rigid-like elastic behavior in a particular component is enforced by using a stiff, linear elastic spring in that component. The stiffness of the spring is chosen automatically and depends on the circumstances in which the connector is used. In Abaqus/Standard the stiffness is taken to be 10 times larger than the average stiffness of the surrounding elements to which the connector element attaches. If the average stiffness cannot be computed (as would be the case when the connector element does not attach to other elements or attaches to rigid bodies), a stiffness of 10^6 is used. In Abaqus/Explicit a Courant stiffness is first computed by considering the average mass at the connector element nodes and the stable time increment in the analysis. In most cases the Courant stiffness is then used to calculate the value of the rigid-like elastic behavior using heuristics that depend on modeling circumstances and the precision (single or double) of the analysis. For example, if plasticity is defined in the connector, the rigid-like elastic stiffness in components involved in the plasticity definition does not exceed one thousandth of the initial yield value. If plasticity is not defined, the rigid-like stiffness is computed as a multiple of the Courant stiffness.

In most cases, the heuristics used in the computation of the rigid-like stiffness produces a stiffness value that is adequate. If this stiffness does not serve the needs of your application, you can always customize the elastic stiffness by specifying the linear stiffness value directly.

Due to the different stiffness values used for rigid-like elastic behavior in Abaqus/Standard and Abaqus/Explicit, you may notice a discontinuity in the behavior when such a model is imported from one solver to the other.

Defining Elastic Connector Behavior in Linear Perturbation Procedures

Available components of relative motion with connector elasticity use the linearized elastic stiffness from the base state. In direct-solution steady-state dynamic and subspace-based steady-state dynamic analyses, the linear elastic stiffness defined by an uncoupled connector elasticity behavior may be frequency dependent.

Output

The Abaqus output variables available for connectors are listed in *Abaqus/Standard Output Variable Identifiers* and *Abaqus/Explicit Output Variable Identifiers*. The following output variables are of particular interest when defining elasticity in connectors:

CU	Connector relative displacements/rotations.
CUE	Connector elastic displacements/rotations.
CEF	Connector elastic forces/moments.

References

- Genta, G., Dynamics of Rotating Systems, Springer, 2005.

Connector Damping Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Connectors*
- *Connector Behavior*
- *Connection-Type Library*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR DAMPING*
- *Defining damping*

Overview

Connector damping behavior:

- can be of a dashpot-like viscous nature in transient or steady-state dynamic analyses;
- can be of a "structural" nature, related to complex stiffness, for steady-state dynamics procedures that support nondiagonal damping;
- can be defined in any connector with available components of relative motion;
- can be specified for each available component of relative motion independently, in which case the behavior can be linear or nonlinear for viscous nature damping;
- can be specified as dependent on relative positions or constitutive motions in several local directions for viscous nature damping; and
- can be specified for all available components of relative motion as coupled damping behavior.

The directions in which the forces and moments act and the relative velocities are measured are determined by the local directions as described in *Connection-Type Library* for each connection type. In dynamic analysis the relative velocities are obtained as part of the integration operator; in quasi-static analysis in Abaqus/Standard the relative velocities are obtained by dividing the relative displacement increments by the time increment.

Defining Linear Uncoupled Viscous Damping Behavior

In the simplest case of linear uncoupled damping you define the damping coefficients for the selected components (that is, C_{11} for component 1, C_{22} for component 2, etc.), which are used in the equation

$$F_i = C_{ii} v_i \quad (\text{no sum on } i),$$

where F_i is the force or moment in the i^{th} component of relative motion and v_i is the velocity or angular velocity in the i^{th} direction. The damping coefficient can depend on frequency (in Abaqus/Standard), temperature, and field variables. See *Input Syntax Rules* for further information about defining data as functions of frequency, temperature, and field variables.

In most cases if frequency-dependent damping behavior is specified in an Abaqus/Standard analysis procedure, the data at zero frequency is used. The exceptions are direct-solution steady-state dynamics, subspace-based steady-state dynamics, and natural or complex eigenvalue extraction.

Input File Usage: Use the following options to define linear uncoupled damping connector behavior:

*CONNECTOR BEHAVIOR, NAME=*name*
*CONNECTOR DAMPING, COMPONENT=*component number*, DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damping**: **Definition: Linear**, **Force/Moment:** *component or components*, **Coupling: Uncoupled**

Defining Linear Coupled Viscous Damping Behavior

In the linear coupled case you define the damping coefficient matrix components, C_{ij}, which are used in the equation

$$F_i = \sum_j C_{ij} v_j,$$

where F_i is the force in the i^{th} component of relative motion, v_j is the velocity in the j^{th} component, and C_{ij} is the coupling between the i^{th} and j^{th} components. The *C* matrix is assumed to be symmetric, so only the upper triangle of the matrix is specified. In connectors with kinematic constraints the entries that correspond to the constrained components of relative motion will be ignored. The damping coefficient can depend on temperature and field variables. See *Input Syntax Rules* for further information about defining data as functions of temperature and field variables.

Input File Usage: Use the following options to define linear coupled damping connector behavior:

*CONNECTOR BEHAVIOR, NAME=*name*
*CONNECTOR DAMPING, DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damping**: **Definition: Linear**, **Force/Moment:** *component or components*, **Coupling: Coupled**

Defining Unsymmetric Linear Coupled Viscous Damping Behavior

As with linear coupled elastic behavior (*Connector Elastic Behavior*), Abaqus/Standard allows you to define an unsymmetric coupled viscous damping matrix. In the linear coupled case you define the damping coefficient matrix components, C_{ij}, which are used in the equation

$$F_i = \sum_j C_{ij} v_j,$$

where F_i is the force in the i^{th} component of relative motion, v_j is the velocity in the j^{th} component, and C_{ij} is the coupling between the i^{th} and j^{th} components. The *C* matrix is assumed to be unsymmetric, so the entire matrix is specified. The entries that correspond to the constrained components of relative motion are ignored. When the unsymmetric matrix storage and solution scheme are used, the damping coefficients can depend on frequency, temperature, and field variables. See *Input Syntax Rules* for further information about defining data as functions of frequency, temperature and field variables.

Input File Usage: Use the following options to define unsymmetric linear coupled viscous damping connector behavior:

*CONNECTOR BEHAVIOR, NAME=*name*
*CONNECTOR DAMPING, UNSYMM,
FREQUENCY DEPENDENCE=ON

Abaqus/CAE Usage: Unsymmetric linear coupled viscous damping behavior is not supported in Abaqus/CAE.

Defining Nonlinear Viscous Damping Behavior

For nonlinear damping you specify forces or moments as nonlinear functions of the velocity in the available components of relative motion directions, $F_i\left(v_1, v_2, \ldots\right)$. These functions can also depend on temperature and field variables. See *Input Syntax Rules* for further information about defining data as functions of temperature and field variables.

Defining Nonlinear Viscous Damping Behavior That Depends on One Component Direction

By default, each nonlinear force or moment function is dependent only on the velocity in the direction of the specified component of relative motion.

Input File Usage: Use the following options:

*CONNECTOR BEHAVIOR, NAME=*name*
*CONNECTOR DAMPING, COMPONENT=*component number*,
NONLINEAR, DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damping**: **Definition: Nonlinear**, **Force/Moment:** *component or components*, **Coupling: Uncoupled**

Defining Nonlinear Viscous Damping Behavior That Depends on Several Component Directions

Alternatively, the functions can depend on the relative positions or constitutive displacements/rotations in several component directions, as described in *Defining Nonlinear Connector Behavior Properties to Depend on Relative Positions or Constitutive Displacements/Rotations*.

Input File Usage: Use the following options to define nonlinear damping connector behavior that depends on components of relative position:

*CONNECTOR BEHAVIOR, NAME=*name*
*CONNECTOR DAMPING, COMPONENT=*component number*,
NONLINEAR, INDEPENDENT COMPONENTS=POSITION,
DEPENDENCIES=*n*

Use the following options to define nonlinear damping connector behavior that depends on components of constitutive displacements or rotations:

*CONNECTOR BEHAVIOR, NAME=*name*
*CONNECTOR DAMPING, COMPONENT=*component number*,
NONLINEAR, INDEPENDENT COMPONENTS=CONSTITUTIVE MOTION,
DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damping**: **Definition: Nonlinear**, **Force/Moment:** *component or components*, **Coupling: Coupled on position** or **Coupled on motion**

Example

Refer to the example in *Figure 1*.

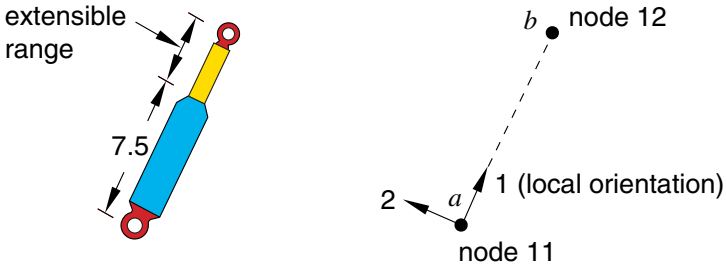


Figure 1: Simplified connector model of a shock absorber.

In addition to the torsional spring resisting relative rotations, the shock absorber damps translational motion along the line of the shock with a dashpot. To include a nonlinear dashpot behavior that is dependent on the relative position between the attachment points, use the following input:

```
*CONNECTOR BEHAVIOR, NAME=sbehavior
...
*CONNECTOR DAMPING, COMPONENT=1,
 INDEPENDENT COMPONENTS=POSITION, NONLINEAR
1
1500.0, 0.1, 0.0
1625.0, 0.2, 0.0
1750.0, 0.1, 10.0
1925.0, 0.2, 10.0
```

Defining Linear Structural Damping Behavior

Structural connector damping is supported in steady-state dynamics and modal transient procedures that support nondiagonal damping (for example, direct solution steady-state dynamics).

Defining Linear Uncoupled Structural Damping Behavior

You define the damping coefficients, s_{jj}, for the selected components (i.e., s_{11} for component 1, s_{22} for component 2, etc.), which are used in the equation

$$F_j = iD_{jj}u_j,$$

where

$$D_{jj} = s_{jj}K_{jj} \quad (\text{no sum on } j)$$

is the structural damping matrix, F_j is the imaginary part of the force or moment in the j^{th} direction of relative motion, u_j is the displacement in the j^{th} direction, and K_{jj} is the stiffness matrix. The damping coefficient can depend on frequency.

Input File Usage: Use the following options:

*CONNECTOR BEHAVIOR, NAME=*name*
*CONNECTOR DAMPING, COMPONENT=*component number*,
TYPE=STRUCTURAL

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damping**: **Damping type: Structural**, **Definition: Linear**, **Force/Moment:** *component or components*, **Coupling: Uncoupled**

Defining Linear Coupled Structural Damping Behavior

You define 21 s_{lj} damping coefficients (the symmetric half of the 6×6 damping coefficient matrix), which are used in the equation

$$F_l = iD_{lj}u_j,$$

where

$$D_{lj} = is_{lj}K_{lj} \quad (\text{no sum on } l, j)$$

is the structural damping matrix, F_l is the imaginary part of the force in the l^{th} direction of relative motion, u_j is the displacement in the j^{th} direction, and K_{lj} is the stiffness matrix. The damping coefficient matrix cannot depend on frequency.

Input File Usage: Use the following options:

*CONNECTOR BEHAVIOR, NAME=*name*
*CONNECTOR DAMPING, TYPE=STRUCTURAL

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damping**: **Damping type: Structural**, **Definition: Linear**, **Force/Moment:** *component or components*, **Coupling: Coupled**

Defining Connector Damping Behavior in Linear Perturbation Procedures

In both the direct-solution and subspace-based steady-state dynamic procedures, the viscous or structural damping defined using an uncoupled connector damping behavior may be frequency dependent. In other linear perturbation procedures connector damping behavior is ignored.

Output

The Abaqus output variables available for connectors are listed in *Abaqus/Standard Output Variable Identifiers* and *Abaqus/Explicit Output Variable Identifiers*. The following output variables are of particular interest when defining damping in connectors:

CV Connector relative velocities/angular velocities.

CVF Connector viscous forces/moments.

Connector Functions for Coupled Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Connectors*
- *Connector Friction Behavior*
- *Connector Plastic Behavior*
- *Connector Damage Behavior*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR DERIVED COMPONENT*
- **CONNECTOR POTENTIAL*
- *Specifying connector derived components*
- *Specifying potential terms*

Overview

This section describes how to define two special functions used to specify complex coupled behavior for a connector element in Abaqus: derived components and potentials.

Connector derived components are user-specified component definitions based on a function of intrinsic (1 through 6) connector components of relative motion. They can be used:

- to specify the friction-generating normal force in connectors as a complex combination of connector forces and moments, and
- as an intermediate result in a connector potential function.

Connector potentials are user-defined functions of intrinsic components of relative motion or derived components. These functions can be quadratic, elliptical, or maximum norms. They can be used to define:

- the yield function for connector coupled plasticity when several available components of relative motion are involved simultaneously,
- the potential function for coupled user-defined friction when the slip direction is not aligned with an available component of relative motion,
- a magnitude measure as a coupled function of connector forces or motions used to detect the initiation of damage in the connector, and
- an effective motion measure as a coupled function of connector motions to drive damage evolution in the connector.

Defining Derived Components for Connector Elements

The definition of coupled behavior in connector elements beyond simple linear elasticity or damping often requires the definition of a resultant force involving several intrinsic (1 through 6) components or the definition of a “direction” not aligned with any of the intrinsic components. These user-defined resultants or directions are called derived components. The forces and motions associated with these derived components are functions of the forces and motions in the intrinsic relative components of motion in the connector element.

The following discussion presents a few conceptual examples of situations where derived components can be useful. The examples are not meant to represent the actual implementation of specific connector types in Abaqus, which are discussed in more details in later sections.

Consider the case of a SLOT connector for which frictional effects (see *Connector Friction Behavior*) are defined in the only available component of relative motion (the 1-direction). The two constraints enforced by this connection type will produce two reaction forces (f_2 and f_3), as shown in *Figure 1*. Both forces generate friction in the 1-direction in a coupled fashion.

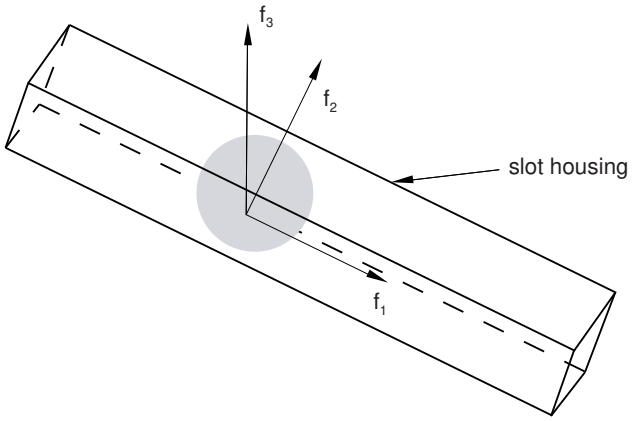


Figure 1: Resultant contact force in a SLOT connector.

A reasonable estimate for the resulting contact force is

$$F^{\text{contact}}_{\text{derived}} = g(\mathbf{f}) = \sqrt{f_2^2 + f_3^2},$$

where $\mathbf{f}$ is the collection of connector forces and moments in the intrinsic components. The function $g(\mathbf{f})$ can be specified as a derived component.

Resultant forces that can be defined as derived components may take more complicated forms. Consider a BUSHING connection type for which a tensile (Mode I) damage mechanism with failure is to be specified in the 1-direction. You may wish to include the effects of the axial force f_1 and of the resultant of the "flexural" moments m_2 and m_3 in defining an overall resultant force in the axial direction upon which damage initiation (and failure) can be triggered, as shown in *Figure 2*.

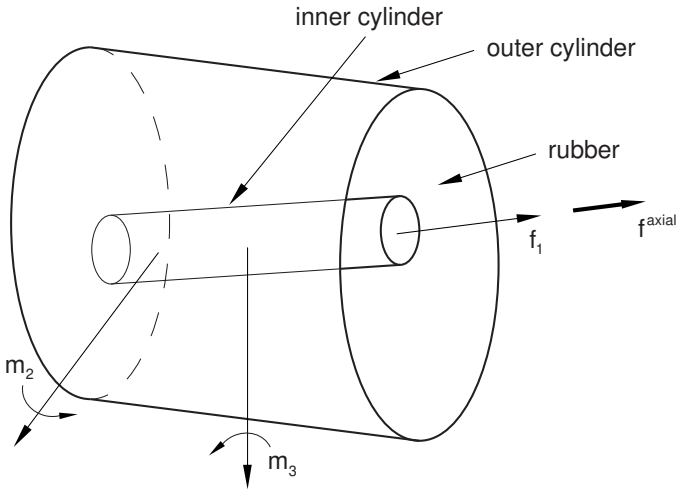


Figure 2: Resultant axial force in a BUSHING connector.

One choice would be to define the resultant axial force as

$$F^{\text{axial}}_{\text{derived}} = g(\mathbf{f}) = |f_1| + \alpha\sqrt{(m_2^2 + m_3^2)},$$

where α is a geometric factor relating translations to rotations with units of one over length. The function $g(\mathbf{f})$ can be specified as a derived component.

A derived component can also be interpreted as a user-specified direction that is not aligned with the connector component directions. For example, if the motion-based damage with failure criterion in a CARTESIAN connection with elastic behavior does not align with the intrinsic component directions, the damage criterion can be defined in terms of a derived component representing a different direction, as shown in *Figure 3*.

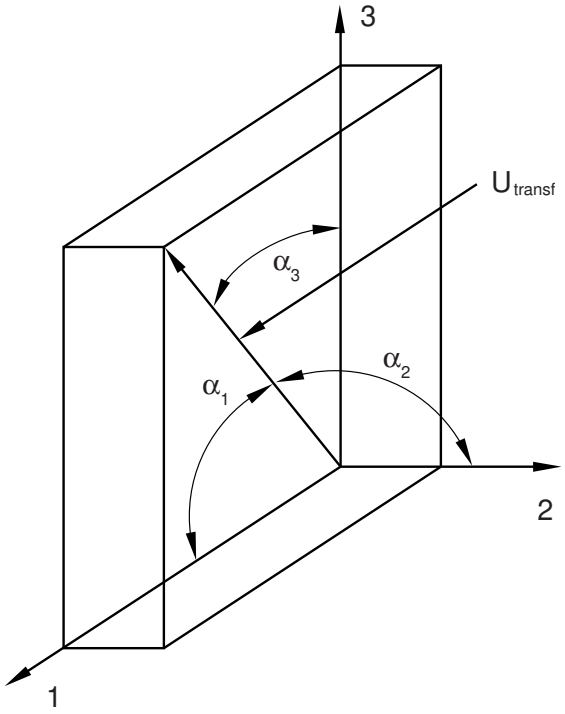


Figure 3: User-defined direction in a CARTESIAN connector.

One possible choice for the direction could be

$$u^{\text{transf}}_{\text{derived}} = g(\mathbf{u}) = a_1 u_1 + a_2 u_2 + a_3 u_3,$$

where $\mathbf{u}$ is the collection of connector relative motions in the components and a_1, a_2, and a_3 can be interpreted as direction cosines ($cos(\alpha_1)$, $cos(\alpha_2)$, $cos(\alpha_3)$). The function $g(\mathbf{u})$ can be specified as a derived component.

Functional Form of the Derived Component

The functional form of a derived component (g) in Abaqus is quite general; you specify its exact form. The derived component is specified as a sum of terms

$$g(\mathbf{c}) = \sum_{i=1}^{N_T} T_i(\mathbf{c}),$$

where $\mathbf{c}$ is a generic name for the connector intrinsic component values (such as forces, $\mathbf{f}$, or motions, $\mathbf{u}$), T_i is the i^{th} term in the sum, and N_T is the number of terms. The appropriate component values for $\mathbf{c}$ are selected

depending on the context in which the derived component is used. T_i is also a summation of several contributions and can take one of the following three forms:

- a norm (g_N-type)

$$T_i(\mathbf{c}) = s_i \sqrt{\sum_{j=1}^{N_c} (\alpha_j c_j)^2}, \qquad \text{or}$$

- a direct sum (g_S-type)

$$T_i(\mathbf{c}) = s_i \sum_{j=1}^{N_c} \alpha_j c_j, \qquad \text{or}$$

- a Macauley sum (g_M-type)

$$T_i(\mathbf{c}) = s_i \sum_{j=1}^{N_c} < \alpha_j c_j >,$$

where s_i is the term's sign (plus or minus), α_j are scaling factors, c_j is the j^{th} component of $\mathbf{c}$, and $< X >$ is the Macauley bracket ($\{X\} = 0 \quad \text{if} \quad X \leq 0 \quad \text{and} = X \quad \text{if} \quad X > 0$). In general, the units of the scaling factors α_j depend on the context. In most cases they are either dimensionless, have units of length, or have units of one over length. The scaling factors should be chosen such that all the terms in the resulting derived component have the same units, and these units must be consistent with the use of the derived component later on in a connector potential or connector contact force.

Defining a Derived Component with Only One Term ($N_T = 1$)

Connector derived components are identified by the names given to them. If one term (T_1) is sufficient to define the derived component *g*, specify only one connector derived component definition.

Input File Usage: **CONNECTOR DERIVED COMPONENT*,
NAME=*derived_component_name*

Abaqus/CAE Usage: Connector derived component names are not supported in Abaqus/CAE; you define the individual derived component terms.

Use the following input to define a connector derived component term for a friction-generating user-defined contact force:

Interaction module: connector section editor: **Add**->**Friction**: **Friction model**: **User-defined**, **Contact Force**, **Specify component**: **Derived component**, click **Edit** to display the derived component editor: click **Add** and select components

Use the following input to define a connector derived component term as an intermediate result in a connector potential function:

Interaction module: connector section editor: **Add**->**Friction**, **Plasticity**, or **Damage**: potential contribution editors: **Specify derived component**, click **Edit** to display the derived component editor: click **Add** and select components

Defining a Derived Component Containing Multiple Terms ($N_T > 1$)

If several terms (T_1, T_2, etc.) are needed in the overall definition of the derived component g, you must define the individual terms.

Input File Usage:

You must specify N_T connector derived component definitions with the same name to define the individual terms. All definitions with the same name will be summed together to produce the desired derived component g. See the spot weld example below for an illustration.

**CONNECTOR DERIVED COMPONENT*, NAME=*derived_component_name*
**CONNECTOR DERIVED COMPONENT*, NAME=*derived_component_name*
...

Abaqus/CAE Usage:

Connector derived component names are not supported in Abaqus/CAE; you define the individual derived component terms.

Interaction module: derived component editor: click **Add** and select components. Repeat, adding terms as necessary.

Specifying a Term in the Derived Component as a Norm

By default, a derived component term is computed as the square root of the sum of the squares of each intrinsic component contribution. If the term has only one contribution ($N_C = 1$), the norm has the same meaning as the absolute value.

Input File Usage:

**CONNECTOR DERIVED COMPONENT*, NAME=*derived_component_name*, OPERATOR=NORM (default)

For example, the following input can be used to define the $F^{\text{axial}}_{\text{derived}}$ component discussed above:

**CONNECTOR DERIVED COMPONENT*, NAME=axial
1
1.0,

** $T_1 = \sqrt{(1.0 * f_1)^2} = |1.0 * f_1|$
**CONNECTOR DERIVED COMPONENT*, NAME=axial
5, 6
α, α

** $T_2 = \sqrt{(\alpha m_2)^2 + (\alpha m_3)^2}$

The `axial` derived component is $F^{\text{axial}}_{\text{derived}} = T_1 + T_2$.

Abaqus/CAE Usage:

Interaction module: derived component editor: **Add**: **Term operator: Square root of sum of squares**

Specifying a Term in the Derived Component as a Direct Sum

Alternatively, you can choose to compute a derived component term as the direct sum of the intrinsic component contributions.

Input File Usage: **CONNECTOR DERIVED COMPONENT*, NAME=*derived_component_name*, OPERATOR=SUM

For example, the following input can be used to define the $u_{\text{derived}}^{\text{transf}}$ component discussed above:

**CONNECTOR DERIVED COMPONENT*, NAME=transf,
OPERATOR=SUM
1, 2, 3
a_1, a_2, a_3
** $T_1 = a_1 u_1 + a_2 u_2 + a_3 u_3$

The `transf` derived component is $u_{\text{derived}}^{\text{transf}} = T_1$.

Abaqus/CAE Usage: Interaction module: derived component editor: **Add**: **Term operator: Direct sum**

Specifying a Term in the Derived Component as a Macauley Sum

Alternatively, you can choose to compute a derived component term as the Macauley sum of the intrinsic component contributions.

Input File Usage: **CONNECTOR DERIVED COMPONENT*, NAME=*derived_component_name*, OPERATOR=MACAULEY SUM

For example, the following input can be used to define the first term of the normal component of the force (F_n) in the spotweld example discussed below:

**CONNECTOR DERIVED COMPONENT*, NAME=normal,
OPERATOR=MACAULEY SUM
3
1.0
** $T_1 =< f_3 >$

Abaqus/CAE Usage: Interaction module: derived component editor: **Add**: **Term operator: Macauley sum**

Specifying the Sign of a Term

You can specify whether the sign of a derived component term should be positive or negative.

Input File Usage: Use one of the following options:

**CONNECTOR DERIVED COMPONENT*, NAME=*derived_component_name*, SIGN=POSITIVE (default)
**CONNECTOR DERIVED COMPONENT*, NAME=*derived_component_name*, SIGN=NEGATIVE

Abaqus/CAE Usage: Interaction module: derived component editor: **Add**: **Overall term sign: Positive** or **Negative**

Defining the Derived Component Contributions to Depend on Local Directions

The scaling factors α_j used in the definition of the derived component can depend on the relative positions or constitutive displacements/rotations in several component directions, as described in *Defining Nonlinear Connector Behavior Properties to Depend on Relative Positions or Constitutive Displacements/Rotations*. See the first example in *Connector Friction Behavior* for an illustration.

Input File Usage: Use the following option to define a connector derived component that depends on components of relative position:

**CONNECTOR DERIVED COMPONENT*, INDEPENDENT COMPONENTS=POSITION

Use the following option to define a connector derived component that depends on components of constitutive displacements or rotations:

**CONNECTOR DERIVED COMPONENT*, INDEPENDENT COMPONENTS=CONSTITUTIVE MOTION

Abaqus/CAE Usage: Interaction module: derived component editor: **Add**: **Use local directions: Independent position components** or **Independent constitutive motion components**

Requirements for Constructing a Derived Component Used in Plasticity or Friction Definitions

When a derived component is used to construct the yield function for a plasticity or friction definition, the following simple requirements must be satisfied:

- All N_T terms of a derived component must be of a compatible type (see *Functional Form of the Derived Component*); norm-type terms (g_N-type) cannot be mixed with direct sum-type terms (g_S-type) in the same derived component definition but can be mixed with Macauley sum-type terms (g_M-type).
- If all N_T terms are norm-type terms, the sign of each term must be positive (the default).

If N_T is greater than 1, the associated functions (potentials) in which the derived component is used may become nonsmooth. More precisely, the normal to the hypersurface defined by the potential may experience sudden changes in direction at certain locations. In these cases, Abaqus will automatically smooth-out the defined functions by slightly changing the derived component functional definition. These changes should be transparent to the user as the results of the analysis will change only by a small margin.

Example: Spot Weld

The spot weld shown in *Figure 4* is subjected to loading in the *F*-direction.

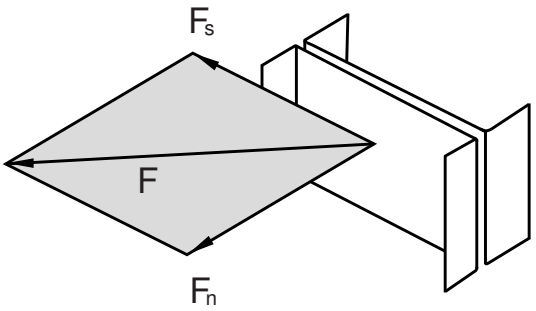


Figure 4: Loading of a spot weld connection.

The connector chosen to model the spot weld has six available components of relative motion: three translations (components 1–3) and three rotations (components 4–6). You have chosen this connection type because you are modeling a general deformation state. However, you would like to define inelastic behavior in the connection in terms of a normal and a shear force, as shown in *Figure 5*, since experimental data are available in this format.

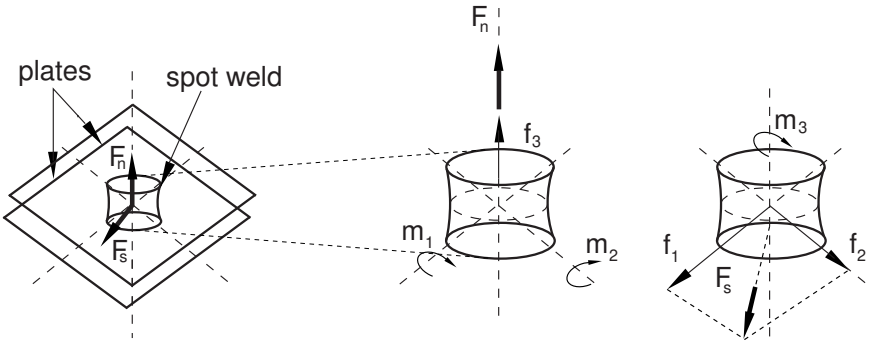


Figure 5: Spot weld connection: derived component definitions.

Therefore, you want to derive the normal and shear components of the force, for example, as follows:

$$F_n = g_n(\mathbf{f}) =< f_3 > + (1/r_n)\sqrt{{m_1}^2 + {m_2}^2},$$

$$F_s = g_s(\mathbf{f}) = (1/r_s)\ |m_3| + \sqrt{{f_1}^2 + {f_2}^2}.$$

In these equations r_n and r_s have units of length; their interpretation is relatively straightforward if you consider the spot weld as a short beam with the axis along the spot weld axis (3-direction). If the average cross-section area of the spot weld is A and the beam's second moment of inertia about one of the in-plane axes is I_{11} (or I_{22}), r_n can be interpreted as the square root of the ratio I_{11}/A (or I_{22}/A). Furthermore, if the cross-section is considered to be circular, r_n becomes equal to a fraction of the spot weld radius. In all cases r_s can be taken to be $2r_n$.

The reasoning above for the interpretation of the calibration constants in the equations is only a suggestion. In general, any combination of constants that would lead to good comparisons with other results (experimental, analytical, etc.) is equally valuable.

To define F_n, you would specify the following two connector derived component definitions, each with the same name:

```
*PARAMETER
Ixx=30.68
A=19.63
```

```
r_n=sqrt(I_xx/A)
r_s=2.0*r_n
or_n = (1/r_n)
or_s = (1/r_s)
*CONNECTOR DERIVED COMPONENT, NAME=normal, OPERATOR=MACAULEY SUM
3
1.0
*CONNECTOR DERIVED COMPONENT, NAME=normal
4, 5
< or_n >, < or_n >
```

The $<\ >$ symbols denote that or_n is specified using a parameter definition. The normal force derived component F_n is defined as the sum of two terms, $g_n(\mathbf{f}) = T_1(\mathbf{f}) + T_2(\mathbf{f})$. The first connector derived component defines the first term $T_1 =< f_3 >$, while the second defines the second term $T_2 = (1/r_n)\sqrt{m_1^2 + m_2^2}$.

Similarly, to define F_s, you would specify the following two connector derived component definitions for the component `shear`:

```
*CONNECTOR DERIVED COMPONENT, NAME=shear
6
< or_s >
*CONNECTOR DERIVED COMPONENT, NAME=shear
1, 2
1.0, 1.0
```

Defining Connector Potentials

Connector potentials are user-defined mathematical functions that represent yield surfaces, limiting surfaces, or magnitude measures in the space spanned by the components of relative motion in the connector. The functions can be quadratic, general elliptical, or maximum norms. The connector potential does not define a connector behavior by itself; instead, it is used to define the following coupled connector behaviors:

- friction,
- plasticity, or
- damage.

Consider the case of a SLIDE-PLANE connection in which frictional sliding occurs in the connection plane, as shown in *Figure 6*.

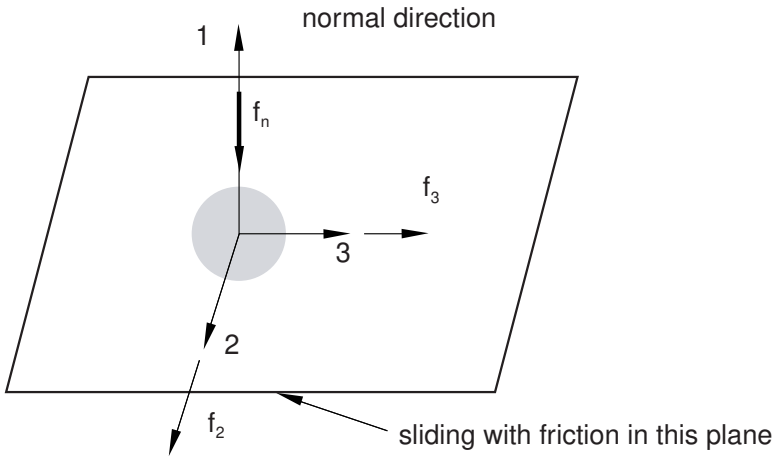


Figure 6: Friction in the SLIDE-PLANE connection.

The function governing the stick-slip frictional behavior (see *Connector Friction Behavior*) can be written as

$$\phi_{fric}\left(\mathbf{f}\right)=\mathbf{P}\left(\mathbf{f}\right)-\mu F_N,$$

where $\mathbf{P}\left(\mathbf{f}\right)$ is the connector potential defining the pseudo-yield function (the magnitude of the frictional tangential tractions in the connector in a direction tangent to the connection plane on which contact occurs), $\mathbf{F_N}$ is the friction-producing normal (contact) force, and μ is the friction coefficient. Frictional stick occurs if $\phi_{fric}<0$, and sliding occurs if $\phi_{fric}=0$. In this case the potential can be defined as the magnitude of the frictional tangential tractions,

$$\mathbf{P}\left(\mathbf{f}\right)=\sqrt{f_2^2+f_3^2}.$$

Connector potentials can also be useful in defining connector damage with a force-based coupled damage initiation criterion. For example, in a connection type with six available components of relative motion you could define a potential

$$\mathbf{P}\left(\mathbf{f}\right)=\sqrt{\left(\frac{f_1}{\alpha_1}\right)^2+\left(\frac{f_2}{\alpha_2}\right)^2+\left(\frac{f_3}{\alpha_3}\right)^2+\left(\frac{m_1}{\beta_1}\right)^2+\left(\frac{m_2}{\beta_2}\right)^2+\left(\frac{m_3}{\beta_3}\right)^2}.$$

Damage (with failure) can be initiated when the value of the potential $\mathbf{P}$ is greater than a user-specified limiting value (usually 1.0). The units of the α and β coefficients must be consistent with the units of the final product. For example, if the intended units of $\mathbf{P}\left(\mathbf{f}\right)$ are newtons, the α coefficients are dimensionless while the β coefficients have units of length.

Connector potentials can take more complicated forms. Assume that coupled plasticity is to be defined in a spot weld, in which case a plastic yield criterion can be defined as

$$\phi_{plas}\left(\mathbf{f}\right)=\mathbf{P}\left(\mathbf{f}\right)-F^0,$$

where $\mathbf{P}\left(\mathbf{f}\right)$ is the connector potential defining the yield function and F^0 is the yield force/moment. The potential could be defined as

$$\mathbf{P}\left(\mathbf{f}\right)=\left[\left(\frac{max(F_n,0)}{R_n}\right)^\beta+\left(\frac{|F_s|}{R_s}\right)^\beta\right]^{1/\beta},$$

where F_n and F_s could be the named derived components `normal` and `shear` defined in the example in *Defining Derived Components for Connector Elements* above. If F^0 has units of force and F_n and F_s also have units of force, R_n and R_s are dimensionless.

Input File Usage: **CONNECTOR POTENTIAL*

Abaqus/CAE Usage: Use the following input to define connector potentials for friction behavior:

Interaction module: connector section editor: **Add->Friction**: **Friction model: User-defined**, **Slip direction: Compute using force potential**, **Force Potential**

Use the following input to define connector potentials for plasticity behavior:

Interaction module: connector section editor: **Add->Plasticity**: **Coupling: Coupled**, **Force Potential**

Use the following input to define connector potentials for damage behavior:

Interaction module: connector section editor: **Add->Damage**: **Coupling: Coupled**, **Initiation Potential** or **Evolution Potential**

Functional Form of the Potential

The functional form of the potential $\mathbf{P}$ in Abaqus is quite general; you specify its exact form. The potential is specified as one of the following direct functions of several contributions:

a quadratic form

$$\mathbf{P}(\mathbf{c}) = \left(\sum_{i=1}^{N_p} s_i P_i(\mathbf{c})^2 \right)^{\frac{1}{2}},$$

a general elliptical form

$$\mathbf{P}(\mathbf{c}) = \left(\sum_{i=1}^{N_p} s_i P_i(\mathbf{c})^{\alpha_i} \right)^{\frac{1}{\beta}}, \quad \text{or}$$

a maximum form

$$\mathbf{P}(\mathbf{c}) = \max_{i=1}^{N_p} s_i P_i(\mathbf{c}),$$

where $\mathbf{c}$ is a generic name for the connector intrinsic component values (such as forces, $\mathbf{f}$, or motions, $\mathbf{u}$), P_i is the i^{th} contribution to the potential, N_p is the number of contributions, β and α_i are positive numbers (defaults $\beta = 2.0$, $\alpha_i = \beta$), and s_i is the overall sign of the contribution (1.0 – default, or –1.0). The appropriate component values for $\mathbf{c}$ are selected depending on the context in which the potential is used in. The positive exponents (α_i, β) and the sign s_i should be chosen such that the contribution P_i yields a real number.

P_i is a direct function of either an intrinsic connector component (1 through 6) or a derived connector component. Since derived components are ultimately a function of intrinsic components (see *Defining Derived Components for Connector Elements*), the contribution P_i is ultimately a function of $\mathbf{c}$. P_i is defined as

$$P_i(\mathbf{c}) = H_i(X_i(\mathbf{c})) \qquad (\text{no sum on i}),$$

$$X_i(\mathbf{c}) = \frac{E_i(\mathbf{c}) - a_i}{R_i},$$

where

$H_i(X)$

is the function used to generate the contribution:

- absolute value (default, $|X|$),
- Macauley bracket ($\langle X \rangle = 0 \quad \text{if} \quad X \le 0 \quad \text{and} = X \quad \text{if} \quad X > 0$), or
- identity (X);

E_i

is the value of the identified component (intrinsic or derived);

a_i

is a shift factor (default 0.0); and

R_i

is a scaling factor (default 1.0).

The function $H_i(X)$ can be the identity function only if $\alpha_i = \beta = 1.0$. The units of the various coefficients in the equations above depend on the context in which the potential is used. In most cases the coefficients in the equations above are either dimensionless, have units of length, or have units of one over length. In all cases you must be careful in defining potentials for which the units are consistent.

Defining the Potential as a Quadratic or General Elliptical Form

To define a general elliptical form of the potential, you must specify the inverse of the overall exponent, β. You can also define the exponents α_i if they are different from the default value, which is the specified value of β.

Input File Usage:

To define a quadratic form of the potential, you can omit specifying β since its default value is 2.0. Use the following option:

**CONNECTOR POTENTIAL*

component name or number, R_i, , $H_i(X)$, a_i, s_i
...

Use the following option to define a general elliptical form of the potential:

*CONNECTOR POTENTIAL, OPERATOR=SUM, EXPONENT=β

component name or number, R_i, α_i, $H_i\ (X)$, a_i, s_i
...

Each data line defines one contribution to the potential, P_i. The function $H_i\ (X)$ can be ABS (absolute value and the default), MACAULEY (Macauley bracket), or NONE (identity).

Abaqus/CAE Usage: Interaction module: connector section editor: friction, plasticity, or damage behavior option: **Force Potential**, **Initiation Potential**, or **Evolution Potential**: **Operator: Sum**, **Exponent**: 2 (for quadratic form) or β (for elliptical form), select **Add** and enter data for the potential contribution. Repeat, adding contributions as necessary.

Defining the Potential as a Maximum Form

Alternatively, you can define the potential as a maximum form.

Input File Usage: *CONNECTOR POTENTIAL, OPERATOR=MAX

component name or number, R_i, , $H_i\ (X)$, a_i, s_i
...

Each data line defines one contribution to the potential, P_i. The function $H_i\ (X)$ can be ABS (absolute value and the default), MACAULEY (Macauley bracket), or NONE (identity).

Abaqus/CAE Usage: Interaction module: connector section editor: friction, plasticity, or damage behavior option: **Force Potential**, **Initiation Potential**, or **Evolution Potential**: **Operator: Maximum**, select **Add** and enter data for the potential contribution. Repeat, adding contributions as necessary.

Requirements for Constructing a Potential Used in Plasticity or Friction Definitions

The connector potential, $\mathbf{P}\ (\mathbf{c})$, can be defined using intrinsic components of relative motion, derived components, or both. A particular contribution to the potential may be one of the following two types:

- A norm-type contribution (P_N) defined using the absolute value or the Macauley bracket functions or using a combination of norm-type g_N and Macauley sum-type g_M derived components (see *Requirements for Constructing a Derived Component Used in Plasticity or Friction Definitions*) with any of the available functions.
- A sum-type contribution (P_S) defined using an intrinsic component of relative motion or a derived component of type g_S (see *Requirements for Constructing a Derived Component Used in Plasticity or Friction Definitions*) together with the identity function.

When used in the context of connector plasticity or connector friction, the potential must be constructed such that the following requirements are satisfied:

- All N_p contributions to the potential must be of the same type. Mixed P_N and P_S contributions are not allowed in the same potential definition.

- If all N_T terms are P_N-type terms, the sign of each term must be positive (the default).
- The positive numbers β and α_i cannot be smaller than 1.0 and must be equal (the default).

Example: Spot Weld

Referring to the spot weld shown in *Figure 5* and the yield function $\phi_{plas}(\mathbf{F})$ defined above, you would define this potential using the derived components `normal` and `shear` with the following input:

```
*PARAMETER
R_n=0.02
R_s=0.05
β=1.5
*CONNECTOR POTENTIAL, EXPONENT=β
normal, R_n, , MACAULEY
shear, R_s, , ABS
```

Output

The Abaqus/Explicit output variables available for connectors are listed in *Abaqus/Explicit Output Variable Identifiers*. The following variables (available only in Abaqus/Explicit) are of particular interest when defining connector functions for coupled behavior:

CDERF Connector derived force/moment with the connector derived component name appended to the output variable. If the connector derived component is used with connector plasticity, connector friction, and connector damage initiation (type force), the derived components used to form the potential represent forces and this quantity is available for both field and history output. If connector friction is used with contact force, the derived components are not used to form a potential, and the derived force is in fact the connector normal forceCNF (which is available for connector history output.)

CDERU Connector derived displacement/rotation with the connector derived component name appended to the output variable. If the connector derived component is used with motion type for the connector damage initiation and connector damage evolution, the derived components to form the potential represent displacements and this quantity is available for both field and history output.

Connector Friction Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Connectors*
- *Connection Types*
- *Connector Behavior*
- *Connector Functions for Coupled Behavior*
- **CHANGE FRICTION*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR DERIVED COMPONENT*
- **CONNECTOR FRICTION*
- **CONNECTOR POTENTIAL*
- **FRICTION*
- *Defining friction*

Overview

Frictional effects can be defined in any connector with available components of relative motion. A typical connector might have several pieces that are in relative motion and are contacting with friction. Therefore, both frictional forces and frictional moments may develop in the connector available components of relative motion.

To define connector friction in Abaqus, you must specify the following:

- the friction law as governed by a friction coefficient;
- the contributions to the friction-generating connector contact forces or moments; and
- the local “tangent” direction in which the friction forces/moments act.

The friction coefficient can be

- expressed in a general form in terms of slip rate, contact force, temperature, and field variables;
- defined by a static and kinetic term with a smooth transition zone defined by an exponential curve; and
- limited by a tangential maximum force, F_{max}, which is the maximum value of tangential force that can be carried by the connector before sliding occurs.

Abaqus provides two alternatives for specifying the other aspects of friction interactions in connectors:

- Predefined friction interactions for which you need to specify a set of parameters that are characteristic of the connection type for which friction is modeled. Abaqus automatically defines the contact force contributions and the local “tangent” directions in which friction occurs. Predefined friction interactions represent common cases and are available for many connection types (see *Connection Types*). If desired, known internal contact forces (such as from a press-fit assembly) can be defined as well.
- User-defined friction interactions for which you define all friction-generating contact force contributions and the local “tangent” directions along which friction occurs. The user-defined friction interactions can be used if predefined friction is not available for the connection type of interest or if the predefined friction interaction does not adequately describe the mechanism being analyzed. Although more complicated to utilize, user-defined interactions:
  - are very general in nature due to flexibility in defining arbitrary sliding directions via connector potentials and contact forces via connector derived components;

- allow for the specification of sliding directions, contact forces, and additional internal contact forces as functions of connector relative position or motion, temperature, and field variables (the internal contact forces can also be dependent on accumulated slip); and
- allow for several friction definitions to be used in the same connection applied in different components of relative motion.

Friction Formulation in Connectors

The basic concept of Coulomb friction between two contacting bodies is the relation of the maximum allowable frictional (shear) force across an interface to the contact force between the contacting bodies. In the basic form of the Coulomb friction model, two contacting surfaces can carry shear forces, $\mathbf{F_t}$, up to a certain magnitude across their interface before they start sliding relative to one another; this state is known as sticking. The Coulomb friction model defines this critical shear force as $\mu\mathbf{F_N}$, where μ is the coefficient of friction and F_N is the contact force. The stick/slip calculations determine when a point transitions from sticking to slipping or from slipping to sticking. Mathematically, the relationship can be formalized as

$$\Phi = |\mathbf{F_t}| - \mu\mathbf{F_N} \leq 0.$$

Frictional stick occurs if $\Phi < 0$; and sliding occurs if $\Phi = 0$, in which case the friction force is $\mu\mathbf{F_N}$.

Friction in connectors is based on the analogy that contacting surfaces of various parts inside a connector device transmit tangential as well as normal forces across their interfaces. The normal (contact) forces, $\mathbf{F_N}$, are typically generated by kinematic constraints or by elastic forces/moments in the connector. Connector friction can be used to model tangential (shear) forces, $\mathbf{F_t}$, in the space spanned by the available components of relative motion for both stick and slip conditions. *Figure 1* illustrates the simplest frictional mechanism in connectors, a SLOT connector in a two-dimensional analysis.

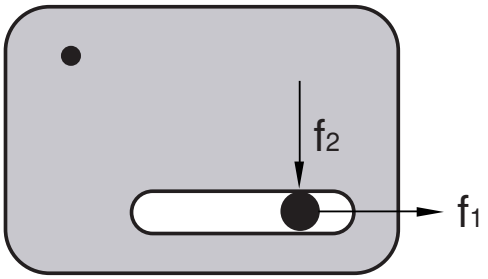


Figure 1: Friction in a two-dimensional SLOT connection.

The local tangent direction in which frictional sliding occurs is the 1-direction (tangential traction $\mathbf{F_t} = f_1$), and the normal force is developed by the kinematic constraint enforcing the SLOT constraint in the 2-direction, $\mathbf{F_N} = f_2$. The friction model is defined in this case by

$$\Phi = |\mathbf{F_t}| - \mu\mathbf{F_N} \leq 0,$$

which in case of slip predicts a friction force $f_1 = \mu f_2$ as expected. In this case the friction model is straightforward to understand as the slip direction is along an intrinsic (1 through 6) component of relative motion and the normal force is given only by the force in one other single component orthogonal to the sliding direction.

In many connectors the definition of the tangential tractions is more complex. For example, friction may develop in a tangent direction that spans two or more available components of relative motion. Consider the case of frictional sliding in a SLIDE-PLANE connection as illustrated in *Connector Functions for Coupled Behavior*.

In this case the friction-generating normal force is given by the constraint force in the 1-direction, $\mathbf{F_N} = f_1$. However, the magnitude of the tangential tractions is given by

$$\mathbf{F}_t = \sqrt{f_2^2 + f_3^2},$$

thus including contributions from two components of relative motion. The instantaneous direction of frictional slip in the 2–3 plane is not known a priori.

In many connectors the normal force may have contributions from several connector components. Consider the case of a three-dimensional SLOT as illustrated in *Connector Functions for Coupled Behavior*. In this case the magnitude of the tangential tractions is given by $\mathbf{F}_t = f_1$, but the normal force is generated by constraint forces in both the 2- and 3-directions and can be written as

$$\mathbf{F}_N = \sqrt{f_2^2 + f_3^2}.$$

In the most general case both the tangential tractions and the normal force may have contributions from several components. Further, the component directions may include both translations (forces) and rotations (moments). Thus, friction modeling in connectors is defined in a more general form, as follows. First, the function Φ governing the stick-slip condition is defined as

$$\Phi = \mathbf{P}(\mathbf{f}) - \mu \mathbf{F}_N \leq 0,$$

where $\mathbf{f}$ is the collection of forces in the connector; $\mathbf{P}(\mathbf{f})$ is the connector potential (see *Connector Functions for Coupled Behavior*), which represents the magnitude of the frictional tangential tractions in the connector in a direction tangent to the surface on which contact occurs; and $\mathbf{F}_N$ is the friction-producing normal (contact) force on the same contact surface. Frictional stick occurs if $\Phi < 0$; and sliding occurs if $\Phi = 0$, in which case the friction force is $\mu \mathbf{F}_N$.

The normal force, $\mathbf{F}_N$, is the sum of a magnitude measure of contact force-producing connector forces, $\mathbf{F}_C = g(\mathbf{f})$, and a self-equilibrated internal contact force (such as from a press-fit assembly), $\mathbf{F}_C^{\text{int}}$:

$$\mathbf{F}_N = \left| \mathbf{F}_C + \mathbf{F}_C^{\text{int}} \right| = \left| g(\mathbf{f}) + \mathbf{F}_C^{\text{int}} \right|.$$

The function $g(\mathbf{f})$ is given by a connector derived component definition as illustrated in *Connector Functions for Coupled Behavior*. Using this formalism, we can easily reconstruct the examples illustrated above:

- In the two-dimensional SLOT case, $\mathbf{P}(\mathbf{f}) = |f_1|$ and $g(\mathbf{f}) = f_2$.
- In the SLIDE-PLANE case, $\mathbf{P}(\mathbf{f}) = \sqrt{f_2^2 + f_3^2}$ and $g(\mathbf{f}) = f_1$.
- In the three-dimensional SLOT case, $\mathbf{P}(\mathbf{f}) = |f_1|$ and $g(\mathbf{f}) = \sqrt{f_2^2 + f_3^2}$.

See the examples at the end of this section for more complex illustrations of friction definitions in connectors.

If frictional effects are defined for a rotational component of relative motion (such as in a HINGE connector), it is often more convenient to define “tangential” moments and “normal” moments instead of tangential tractions/forces and normal forces. The pseudo-yield function governing the stick/slip behavior is defined in a similar fashion:

$$\Phi = \mathbf{P}(\mathbf{f}) - \mu \mathbf{M}_N \leq 0,$$

where the “normal” moment $\mathbf{M}_N$ is written as

$$\mathbf{M}_N = \left| \mathbf{M}_C + \mathbf{M}_C^{\text{int}} \right| = \left| g(\mathbf{f}) + \mathbf{M}_C^{\text{int}} \right|.$$

$\mathbf{M}_{\mathbf{C}}^{\mathbf{int}}$ is the self-equilibrated friction-generating internal "contact" moment (for example, from press fit). See *Specifying Friction in a HINGE Connection* at the end of this section for an illustration.

Predefined Friction Behavior

Predefined friction interactions allow you to model typical frictional mechanisms in commonly used connector types without having to define the mechanics of the frictional response. Instead of specifying the potential, $\mathbf{P}$, directly to define the magnitude measure of the tangential tractions and the contact force $\mathbf{F}_{\mathbf{C}} = \boldsymbol{g}$ via a derived component, you specify:

- a set of friction-related parameters associated with the connection type, which include geometric parameters specific to the connection type and, optionally, the internal contact force $\mathbf{F}_{\mathbf{C}}^{\mathbf{int}}$ or contact moment $\mathbf{M}_{\mathbf{C}}^{\mathbf{int}}$; and
- the friction law (governed by the friction coefficient) as described in *Defining the Friction Coefficient*.

Abaqus then automatically generates internally the potential, $\mathbf{P}$, and the contact force, $\mathbf{F}_{\mathbf{C}}$, based on the connection type and geometric parameters provided. *Table 1* shows the connection types for which predefined friction interactions are available and the associated friction-related parameters. The meanings of the geometric parameters as well as the corresponding potentials and derived components automatically generated by Abaqus are described in *Connection Types*.

Table 1: Predefined friction-related parameters.

Connection type	Friction-related parameters	
	Geometric parameters	Internal contact force/moment
CYLINDRICAL	R, L	$\mathbf{F}_{\mathbf{C}}^{\mathbf{int}}$
HINGE	$\boldsymbol{R_p}$, $\boldsymbol{R_a}$, $\boldsymbol{L_s}$	$\mathbf{M}_{\mathbf{C}}^{\mathbf{int}}$
PLANAR	R	$\mathbf{F}_{\mathbf{C}}^{\mathbf{int}}$, $\mathbf{M}_{\mathbf{C}}^{\mathbf{int}}$
SLIDE-PLANE	None	$\mathbf{F}_{\mathbf{C}}^{\mathbf{int}}$
SLOT	None	$\mathbf{F}_{\mathbf{C}}^{\mathbf{int}}$
TRANSLATOR	$\boldsymbol{R_r}$, L	$\mathbf{F}_{\mathbf{C}}^{\mathbf{int}}$
UJOINT	$\boldsymbol{R_p}$, $\boldsymbol{R_\alpha}$, $\boldsymbol{L_\alpha}$, $\boldsymbol{L_a}$	$\mathbf{M}_{\mathbf{C}_1}^{\mathbf{int}}$, $\mathbf{M}_{\mathbf{C}_3}^{\mathbf{int}}$
SLIPRING	None	None

See the examples at the end of this section for illustrations of predefined friction.

Input File Usage: **CONNECTOR FRICTION*, PREDEFINED
friction-related parameters outlined in Table 1

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Friction**: **Friction model: Predefined**, **Predefined Friction Parameters**, *enter the friction-related parameters outlined in Table 1 in the data table*

User-Defined Friction Behavior

User-defined friction behavior can be used if predefined friction is not available for the connection type of interest or if the predefined friction interaction does not describe adequately the mechanism being analyzed. For user-defined friction you must specify:

- "tangent" direction information, as follows:
  - if the slip direction is known, you specify directly the direction in which friction forces/moments act, from which Abaqus constructs the potential $\mathbf{P}(\mathbf{f})$;
  - if the slip direction is unknown, you specify the potential $\mathbf{P}(\mathbf{f})$ from which Abaqus computes the instantaneous slip direction;
- the friction-producing normal force, $\mathbf{F_N}$, or normal moment, $\mathbf{M_N}$, by defining at least one of the following:
  - the contact force $\mathbf{F_C}$ or contact moment $\mathbf{M_C}$; and/or
  - the internal contact force $\mathbf{F_C^{int}}$ or contact moment $\mathbf{M_C^{int}}$; and
- the friction law (governed by the friction coefficient) as described in *Defining the Friction Coefficient*.

Specifying the Slip Direction Aligned with an Available Component of Relative Motion

The friction tangent direction is identified by specifying an available component (1–6) to define friction forces or moments in a specified intrinsic connector local direction. This is the natural choice in cases when the connector element has only one available component of relative motion (for example, SLOT, REVOLUTE, or TRANSLATOR); in these cases the relative slip between the various parts forming the physical connection occurs in one local direction only. In connections with two or more available components of relative motion, specifying a particular available component of relative motion allows you to specify frictional effects in that direction only, if desired. For example, in the case of a CYLINDRICAL connection, specifying component 1 defines frictional effects only in translation while rotation around the axis is ignored for friction.

Abaqus constructs the potential, $\mathbf{P}(\mathbf{f})$, automatically as

$$\mathbf{P}(\mathbf{f}) = |f_i|,$$

where f_i is the force/moment in the specified component i.

Input File Usage: **CONNECTOR FRICTION*, COMPONENT=*i*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Friction**: **Friction model: User-defined**, **Slip direction: Specify direction**, *component*

Specifying the Potential When the Slip Direction Is Unknown

In connection types with two or more available components of relative motion, frictional slipping is not necessarily solely along one of the available components of relative motion. In such cases the instantaneous slip direction is not known, as illustrated in the SLIDE-PLANE case in *Friction Formulation in Connectors*. Another example is the CYLINDRICAL connection in which frictional sliding occurs in a direction tangent to the cylindrical surface, thus involving simultaneously a translational slip in the local 1-direction and a rotational slip about the same axis (see the first example at the end of this section for an illustration). Thus, frictional slip may occur in a coupled fashion spanning several available components simultaneously.

In such cases you must specify the magnitude measure of the tangential tractions on the assumed contact surface using a connector potential definition, $\mathbf{P}(\mathbf{f})$. Abaqus then computes the instantaneous slip direction simultaneously with the stick-slip determination similar to the surface-based three-dimensional frictional contact computations described in *Coulomb friction*. This procedure is best illustrated for the SLIDE-PLANE case, as follows:

- First, the potential $\mathbf{P}(\mathbf{f}) = \sqrt{f_2^2 + f_3^2}$ is evaluated.
- Slipping occurs if the pseudo-yield function $\Phi \geq 0$.
- The two vector components (the local 2- and 3-directions) of the instantaneous slip direction are given by the ratios of the two shear forces, f_2 and f_3, normalized by the magnitude of the potential.

In general, this strategy is extended to the space spanned by the available components of relative motion associated with the connection type that ultimately participate in the potential definition (see *Connector Functions for Coupled Behavior*). For example, up to two components for SLIDE-PLANE or CYLINDRICAL connections, three components for CARDAN connections, and six components for a user-assembled connection using CARTESIAN and CARDAN connections can be included in the potential. See the examples below for several illustrations.

Input File Usage: Use the following two options to specify coupled user-defined friction:

**CONNECTOR FRICTION*
**CONNECTOR POTENTIAL*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Friction**: **Friction model: User-defined**, **Slip direction: Compute using force potential**, **Force Potential**

Specifying the Contact Force

You specify the friction-generating user-defined contact force, $\mathbf{F}_C = g(\mathbf{f})$, or contact moment, $\mathbf{M}_C = g(\mathbf{f})$, by referring to either an intrinsic component of relative motion number (1 through 6) or a named connector derived component (see *Defining Derived Components for Connector Elements*).

In the latter case the scaling parameters used in the definition of $g(\mathbf{f})$ can be made functions of identified local directions, temperature, and field variables. It is often desirable to include contributions from both connector forces and moments in the definition of the derived component. In these cases the scaling parameters used to define the derived components should have units of length or one over length for meaningful contact force/moment definitions.

Input File Usage: Use the following option to define a contact force for connector friction using an intrinsic connector component:

**CONNECTOR FRICTION*, CONTACT FORCE=*component number (1–6)*

Use the following options to define a contact force for connector friction using a connector derived component:

**CONNECTOR DERIVED COMPONENT*, NAME=*derived_component_name*
**CONNECTOR FRICTION*, CONTACT FORCE=*derived_component_name*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Friction**: **Friction model: User-defined**, **Contact Force**, **Specify component: Intrinsic component** or **Derived component**, *component* or specify derived component

Connector derived component names are not supported in Abaqus/CAE.

Specifying the Internal Contact Force

Internal contact forces such as contact interference may occur in connectors during the physical assembly of the various pieces forming the connector (for example, a press-fit shaft into the sleeve of a CYLINDRICAL connection). When relative motion occurs between the connector parts, these self-equilibrating contact stresses will produce contact forces, $\mathbf{F}_{\mathbf{C}}^{\mathbf{int}}$, or contact moments, $\mathbf{M}_{\mathbf{C}}^{\mathbf{int}}$; see *Friction Formulation in Connectors*.

The internal contact forces/moments are created by specifying a contact force/moment curve (positive values only) as a function of accumulated slip, temperature, and field variables. The accumulated slip is computed as the sum of the absolute values of all slip increments in an instantaneous slip direction. Consequently, the accumulated slip is monotonically increasing for oscillatory or periodic motion and can be used to model dependencies related to wear or heat generation in the connection.

Input File Usage: The internal contact forces limiting curve is defined on the data lines of the **CONNECTOR FRICTION* option.

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Friction**: **Friction model: User-defined**, **Contact Force**, and enter the **Internal Contact Force** in the data table

Specifying the Internal Contact Force to Depend on Local Directions

The internal contact force can also be defined as dependent on either connector relative positions or constitutive relative motions.

Input File Usage: Use the following option to define an internal contact force that depends on components of relative position:

**CONNECTOR FRICTION*, INDEPENDENT COMPONENTS=POSITION

Use the following option to define an internal contact force that depends on components of constitutive displacements or rotations:

**CONNECTOR FRICTION*,
INDEPENDENT COMPONENTS=CONSTITUTIVE MOTION

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Friction**: **Friction model: User-defined**, **Contact Force**, **Use independent components: Position** or **Motion**

Defining the Friction Coefficient

The connector friction definition uses the standard friction model described in *Frictional Behavior* to define the friction coefficient. The anisotropic friction and friction data associated with the second contact direction are ignored for connector elements. If the friction coefficients are not specified or are set to zero, the connector friction has no effect on the connector behavior. If the equivalent shear force/moment limit, f_t^{max}, is specified

(see *Using the Optional Shear Stress Limit*), the limiting friction force μF_N in the pseudo-yield function Φ (see *Friction Formulation in Connectors*) is replaced by $\min(\mu F_N, f_t^{max})$.

Rough, Lagrange, and user-defined friction cannot be used in connector elements.

Input File Usage: Use the following options:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR FRICTION*
**FRICTION*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Friction**: **Tangential Behavior**, **Friction Coefficient**, and enter the **Friction Coeff.** in the data table

Changing the Friction Coefficients during an Abaqus/Standard Analysis

In Abaqus/Standard the friction coefficients can be changed during the analysis as for any analysis including friction (see *Changing Friction Properties during an Abaqus/Standard Analysis* for details).

Controlling the Unsymmetric Solver in Abaqus/Standard

In Abaqus/Standard friction constraints produce unsymmetric terms when the connector nodes are sliding relative to each other. These terms have a strong effect on the convergence rate if frictional stresses have a substantial influence on the overall displacement field and the magnitude of the frictional stresses is highly solution dependent. Abaqus/Standard will automatically use the unsymmetric solution method if the coefficient of friction is greater than 0.2. If desired, you can turn off the unsymmetric solution method as described in *Defining an Analysis*.

Defining the Stick Stiffness

Abaqus determines whether the connector is sticking or slipping in a similar fashion as for all contact interactions (see *Frictional Behavior*), as outlined in *Friction Formulation in Connectors*. If the model is sticking, the elastic stiffness of the response is determined by the optional stick stiffness that is specified as part of the connector friction definition.

If the stick stiffness is not specified, Abaqus will compute a usually appropriate stick stiffness. In Abaqus/Standard a maximum allowable elastic slip length (or angle) is first defined using either the value of the slip tolerance, F_f, together with an automatically computed characteristic length (angle) in the model or the absolute magnitude of the allowable elastic slip, γ_i, to be used in the stiffness method for sticking friction directly (see *Stiffness Method for Imposing Frictional Constraints in Abaqus/Standard*). The elastic stick stiffness is then determined by simply dividing the current connector limiting friction force by this maximum allowable elastic slip length (angle). In Abaqus/Explicit the elastic stick stiffness is determined from the Courant (stability) condition.

Input File Usage: **CONNECTOR FRICTION*, STICK STIFFNESS=*elastic stiffness*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Friction**: **Stick stiffness: Specify**: *elastic stiffness*

Using Multiple Connector Friction Definitions

Multiple connector frictions can be used as part of the same connector behavior definition. However, only one connector friction definition can be used to define friction interactions for each available component of relative

motion. If predefined friction is used, only one connector friction definition can be associated with a connector behavior definition. At most one coupled user-defined friction definition can be associated with a connector behavior definition. Additional connector friction definitions are permitted for the same connector behavior definition only if the component relative motion spaces for each definition do not overlap; for example, you could define uncoupled connector friction in components 1, 2, and 6 and coupled connector friction (by defining a potential) using components 3, 4, and 5. All connector friction definitions act in parallel and will be summed if necessary. For a particular connector element there will be as many stick-slip calculations as connector friction definitions. See the examples below.

Examples

The following examples illustrate how to define friction in connector elements.

Equivalent Ways of Specifying Friction Behavior in a CYLINDRICAL Connection

In the example in *Figure 2* assume Coulomb-like friction affects the translational motion along the shock and the rotational motion about the shock axis.

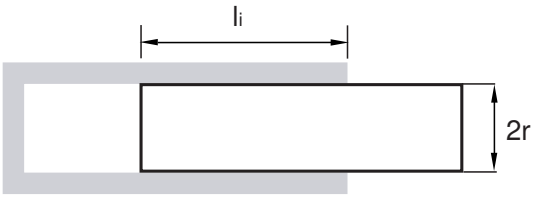


Figure 2: Simplified connector model of a shock absorber.

The coefficient of friction is $\mu = 0.15$, and the overlapping length for the two parts of the shock is $l_i = 0.55$ length units in the undeformed configuration. An average radius of the two cylinders is considered to be $r = 0.24$ units. It is also assumed that the axial motion in the connection is relatively small so that the overlapping length between the connector parts does not change much. The friction-generating contact force has contributions from two sources:

- the normal force from the inner walls pushing against each other (the vector magnitude of the Lagrange multipliers imposing the SLOT constraint), and
- the “bending” in the REVOLUTE constraint (the vector magnitude of the Lagrange multipliers imposing the REVOLUTE constraint).

See *Connection Types* for a detailed discussion of predefined contact forces and tangential tractions in the CYLINDRICAL connection. Two equivalent alternatives to model these frictional effects are shown below:

1. Using the Abaqus predefined friction behavior:

```
*PARAMETER
r=0.24
li=0.55
...
*CONNECTOR FRICTION, PREDEFINED
 <r>, <li>
*FRICTION
  0.15
```

   Using a predefined connector friction behavior yields the most compact definition of frictional effects. This definition requires only the specification of the two friction-relevant geometrical scaling constants.

2. Using a user-defined frictional behavior:

```
*PARAMETER
r=0.24
l_i=0.55
α_1=1.0
α_2i=2.0/l_i
...
*CONNECTOR BEHAVIOR, NAME=shock
*CONNECTOR DERIVED COMPONENT, NAME=normal
2, 3
< α_1 >, < α_1 >
**(sqrt((α_1 * f_2)^2 + (α_1 * f_3)^2))
*CONNECTOR DERIVED COMPONENT, NAME=normal,
5, 6
< α_2i >, < α_2i >
**(sqrt((α_2i * m_2)^2 + (α_2i * m_3)^2))
*CONNECTOR FRICTION, CONTACT FORCE=normal
*CONNECTOR POTENTIAL
1,
4, < r >
*FRICTION
 0.15
```

The contact force “normal” is defined by

$$\mathbf{F_C} = |g(\mathbf{f})| = \sqrt{(\alpha_1 * f_2)^2 + (\alpha_1 * f_3)^2} + \sqrt{(\alpha_{2i} * m_2)^2 + (\alpha_{2i} * m_3)^2}.$$

The connector potential defines the magnitude of the tangential tractions as

$$\mathbf{P}(\mathbf{f}) = \sqrt{f_1^2 + \left(\frac{m_1}{r}\right)^2}.$$

This force magnitude is tangent to the cylindrical surface of the connector on which contact occurs. The choice of normal force definition and potential in this case ensures that the same frictional effects defined in Case A are modeled.

Specifying Friction Interactions in a CYLINDRICAL Connection Accounting for Position Dependencies

In the example in *Figure 2* assume that large axial motion occurs between the two connector parts and, hence, the overlapping length will change significantly during the analysis. For the sake of discussion, assume that the two connector nodes are specified to be overlapped in the initial configuration. Thus, at CP1=0.0 the initial overlap is $l_i = 0.55$ as specified above. If during the analysis the connector relative position along the 1-component reaches CP1=0.45 units, the final overlap would be $l_f = 0.55 - 0.45 = 0.10$. If the connection is subjected to a “bending-like” loading, one can argue that as the overlapping length decreases, the contact forces developed between the two parts become increasingly higher. Use the following user-defined friction behavior definitions to model this dependence of the contact force on relative positions:

```
*PARAMETER
r=0.24
l_i=0.55
l_f=0.1
α_1=1.0
```

```
α2i=2.0/li
α2f=2.0/lf
...
*CONNECTOR BEHAVIOR, NAME=shock
*CONNECTOR DERIVED COMPONENT, NAME=normal
2, 3
< α1 >, < α1 >
**(√((α1 * f2)^2 + (α1 * f3)^2))
*CONNECTOR DERIVED COMPONENT, NAME=normal,
INDEPENDENT COMPONENTS=POSITION
1
5, 6
< α2i >, < α2i >, 0
**(√((α2i * m2)^2 + (α2i * m3)^2) at CP1=0.0)
< α2f >, < α2f >, 0.45
**(√((α2f * m2)^2 + (α2f * m3)^2) at CP1=0.45)
*CONNECTOR FRICTION, CONTACT FORCE=normal
*CONNECTOR POTENTIAL
1,
4, < r >
*FRICTION
 0.15
```

Specifying Friction due to Assembly Contact Interference

Assume a CYLINDRICAL connector element in which the shaft was press-fit into the sleeve, as shown in the initial configuration (relative motion = 0.0) in *Figure 3*.

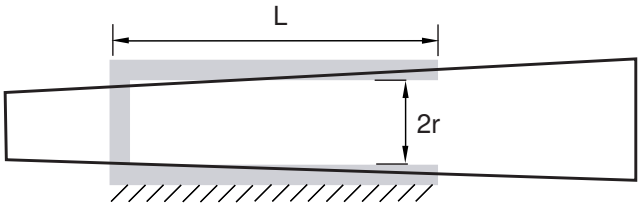


Figure 3: CYLINDRICAL connection with slightly conical pin.

The shaft is not perfectly cylindrical but slightly conical so that its cross-section diameter is increasing in a linear fashion along the shaft direction. If the relative displacement along the shaft direction becomes positive, the contact forces will increase (more contact interference); if the relative displacements become negative (less interference), they will decrease. An exponential decay model is assumed to model the transition from a static coefficient of friction to a kinetic one. Only the positive contact force versus displacement values need to be specified. The following user-defined friction behavior definitions can be used:

```
*PARAMETER
r=0.24
...
*CONNECTOR FRICTION, INDEPENDENT COMPONENTS=CONSTITUTIVE MOTION
1
** (independent component 1)
0.70, -0.7854
0.85, -0.3927
1.0 ,  0.0
```

```
1.15,  0.3927
1.30,  0.7854
*CONNECTOR POTENTIAL
1,
4, <r>...
*FRICTION, EXPONENTIAL DECAY
 0.25, 0.10, 0.2
```

The internal contact forces are specified directly on the data lines to model known contact interference forces as a function of the connector constitutive component of relative motion along component 1. Since no intrinsic component of relative motion number or named connector derived component was specified to define the contact force, the only contribution to the contact force is the specified internal contact force.

Specifying Friction in a HINGE Connection

This example illustrates the use of a connector friction definition to specify frictional effects in a HINGE connection. The friction behavior defines friction moments about the 1-direction, since there are no other available components of relative motion. As illustrated in *HINGE*, the three geometrical scaling constants that need to be specified for predefined friction are the radius of the pin cross-section, R_p=0.12; the effective friction arm in the axial direction, R_a=0.14; and the overlapping length between the pin and the sleeve, L_s=0.65. The friction coefficient is assumed to be μ=0.15. It is assumed that the connector has been assembled with initial known contact interference-producing contact moments of $M_C^{int} = 100.0$ units. The following input could be used to specify the predefined friction behavior in the HINGE connection:

```
*PARAMETER
Rp=0.12
Ra=0.14
Ls=0.65
...
*CONNECTOR FRICTION, PREDEFINED
<Rp>, <Ra>, <Ls>, 100.0
*FRICTION
 0.15
```

Alternatively, a user-defined friction behavior could be specified to define identical frictional effects (see *Connection Types*). Moreover, a reduction of the interference contact forces as the pin wears due to accumulated sliding can be modeled in this case by specifying the internal contact forces/moments to be functions of accumulated slip. The following input can be used:

```
*PARAMETER
Rp=0.12
Ra=0.14
Ls=0.65
α1=Ra
α2=Rp
α3=2.0*Rp/Ls
...
*CONNECTOR DERIVED COMPONENT, NAME=contact_moment
1,
<α1>,
** (√((α1 * f1)^2) = |α1 * f1|)
```

```
*CONNECTOR DERIVED COMPONENT, NAME=contact_moment
2, 3
< α2 >, < α2 >

**(√((α2 * f2)^2 + (α2 * f3)^2))
*CONNECTOR DERIVED COMPONENT, NAME=contact_moment
5, 6
< α3 >, < α3 >

**(√((α3 * m2)^2 + (α3 * m3)^2))
*CONNECTOR FRICTION, COMPONENT=4, CONTACT FORCE=contact_moment
100, 0.0
90 , 1000.0
** interference contact moments decreasing due to wear effects
*FRICTION
0.15
```

The additional friction moments due to contact interference are modeled by specifying decreasing internal contact moments as a function of accumulated rotational slip about the 1-direction. The connector derived component definitions are used to define a contact moment-producing friction in the same direction (component 4). The contact moment is defined by

$$\mathbf{M_C} = |g(\mathbf{f})| = |\alpha_1 f_1| + \sqrt{(\alpha_2 f_2)^2 + (\alpha_2 f_3)^2} + \sqrt{(\alpha_3 m_2)^2 + (\alpha_3 m_3)^2}.$$

The connector potential is defined automatically by Abaqus as $\mathbf{P}(\mathbf{f}) = |m_1|$.

Specifying Friction in a Ball-in-Socket Connection

This example illustrates the specification of frictional effects in a ball-in-socket connection. While the first choice in defining a ball-in-socket connection is JOIN and ROTATION, other rotation parameterizations could be used (JOIN and CARDAN, JOIN and EULER, or JOIN and FLEXION-TORSION). Assuming that the radius of the ball is $R_s = 0.30$ and the coefficient of friction is $\mu = 0.15$, the following lines can be used to define the friction interactions:

```
*PARAMETER
Rs=0.30
...
*CONNECTOR DERIVED COMPONENT, NAME=normal
1, 2, 3
1.0, 1.0, 1.0

**(√((f1)^2 + (f2)^2 + (f3)^2))
*CONNECTOR FRICTION, CONTACT FORCE=normal
*CONNECTOR POTENTIAL
4, < Rs >
5, < Rs >
6, < Rs >
*FRICTION
 0.15
```

The computed connector friction moments and the friction-induced moments at the connector nodes are dependent on the connection type.

Defining Connector Friction Behavior in Linear Perturbation Procedures

Frictional slipping is not allowed in linear perturbation procedures. If a connector is slipping at the end of the last general analysis step, it will slip freely during the current linear perturbation step. Otherwise, Abaqus will allow the connector to slip elastically with the specified stick stiffness or enforce a sticking condition if a stick stiffness is not specified.

Output

The Abaqus output variables available for connectors are listed in *Abaqus/Standard Output Variable Identifiers* and *Abaqus/Explicit Output Variable Identifiers*. The following variables are of particular interest when defining friction in connectors:

CSF
Connector friction forces/moments. In addition to the usual six components associated with connector output variables, CSF includes the scalar CSFC, which is the friction force generated by a coupled friction definition.

CNF
Connector normal forces/moments. CNF includes the scalar CNFC, which is the friction-generating normal force associated with a coupled friction definition.

CASU
Connector accumulated slip. CASU includes the scalar CASUC, which is the accumulated slip associated with a coupled friction definition.

CIVC
Connector instantaneous velocity associated with a coupled friction definition.

Connector Plastic Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Connectors*
- *Connector Behavior*
- *Connector Elastic Behavior*
- *Connector Functions for Coupled Behavior*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR DERIVED COMPONENT*
- **CONNECTOR ELASTICITY*
- **CONNECTOR HARDENING*
- **CONNECTOR PLASTICITY*
- **CONNECTOR POTENTIAL*
- *Defining plasticity*

Overview

Connector plasticity in Abaqus:

- can be used to model plastic/irreversible deformations of parts forming an actual connection device; for example,
  - the pin or the sleeve in a door hinge may deform plastically if the forces/moments acting on them are large enough;
  - connection elements in automotive suspension systems may deform irreversibly due to abusive loading; or
  - spot welds in a car frame and rivets in an airplane could undergo inelastic deformations if the forces acting on the structural members they are a part of are larger than intended;
- is defined in terms of resultant forces and moments in the connector;
- uses perfect plasticity or isotropic/kinematic hardening behavior models;
- can be used when rate-dependent effects are important;
- can be specified in any connectors with available components of relative motion;
- can be used for available components of relative motion for which either elastic or rigid behavior was specified;
- can be used in an uncoupled fashion to define elastic-plastic or rigid plastic response in individual available components of relative motion; and
- can be used to specify coupled elastic-plastic or rigid plastic behavior, in which case the responses in several available components of relative motion are involved simultaneously in a coupled fashion to define plasticity effects.

To define connector plasticity in Abaqus, the following are necessary:

- the elastic or rigid behavior prior to the onset of plasticity;
- a yield function upon which plastic flow will be initiated; and
- hardening behavior to define the initial yield value and, optionally, the yield value evolution after plastic motion initiation.

Plasticity Formulation in Connectors

The plasticity formulation in connectors is similar to the plasticity formulation in metal plasticity (see *Classical Metal Plasticity*). In connectors the stress ($\boldsymbol{\sigma}$) corresponds to the force ($\mathbf{f}$), the strain ($\boldsymbol{\varepsilon}$) corresponds to the constitutive motion ($\mathbf{u}$), the plastic strain ($\boldsymbol{\varepsilon}^{pl}$) corresponds to the plastic relative motion ($\mathbf{u}^{pl}$), and the equivalent plastic strain ($\bar{\varepsilon}^{pl}$) corresponds to the equivalent plastic relative motion ($\bar{u}^{pl}$). The yield function ϕ is defined as

$$\phi\left(\mathbf{f},\bar{\mathbf{u}}^{\mathbf{pl}}\right)=P\left(\mathbf{f}\right)-F^{0}\leq 0,$$

where $\mathbf{f}$ is the collection of forces and moments in the available components of relative motion that ultimately contribute to the yield function; the connector potential, $P\left(\mathbf{f}\right)$, defines a magnitude of connector tractions similar to defining an equivalent state of stress in Mises plasticity and is either automatically defined by Abaqus or user-defined; and F^{0} is the yield force/moment. The connector relative motions, $\mathbf{u}$, remain elastic as long as $\phi<0$; and when plastic flow occurs, $\phi=0$.

If yielding occurs, the plastic flow rule is assumed to be associated; thus, the plastic relative motions are defined by

$$\dot{\mathbf{u}}^{pl}=\dot{\bar{u}}^{pl}\frac{\partial\phi}{\partial\mathbf{f}},$$

where $\dot{\mathbf{u}}^{pl}$ is the rate of plastic relative motion and $\dot{\bar{u}}^{pl}$ is the equivalent plastic relative motion rate.

Loading and Unloading Behavior

Abaqus allows for the following three types of behaviors associated with a plasticity definition when the connector is not actively yielding:

- Linear elastic behavior, shown in *Figure 1*(a), is the most common case since similar behavior can be modeled in metal plasticity, for example, by specifying the Young's modulus. Elastic motion occurs prior to plasticity onset, and unloading from a plastic state occurs on a straight line parallel to the initial loading.

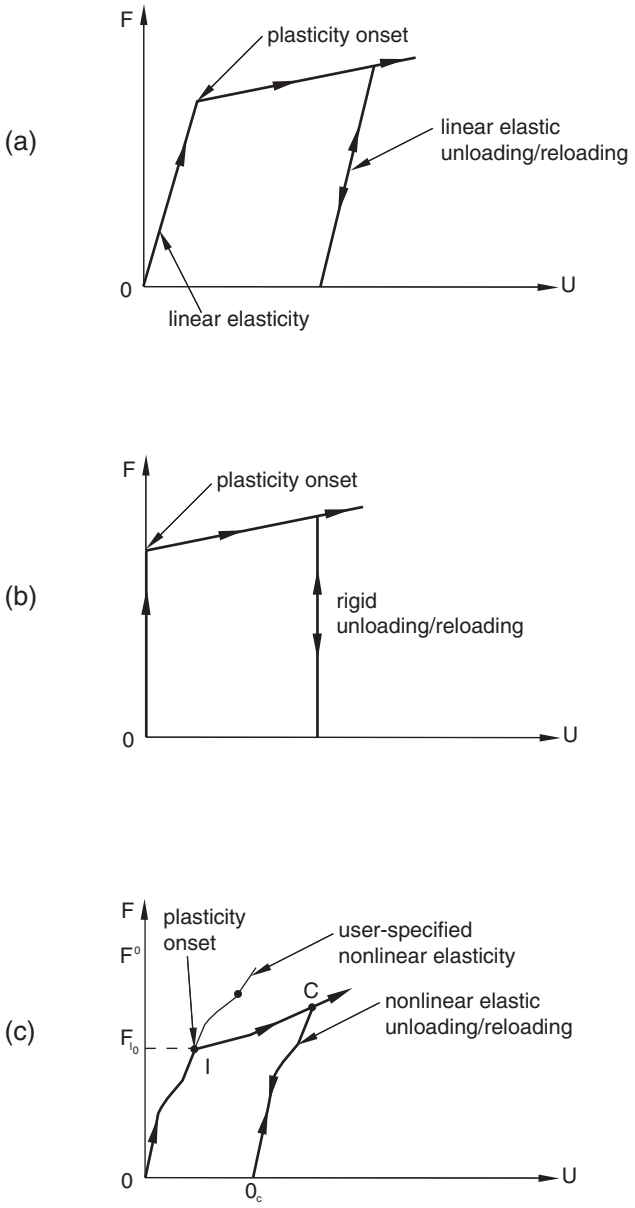


Figure 1: Linear elastic-plastic (a), rigid plastic (b), and nonlinear elastic-plastic (c) response.

- Rigid behavior, shown in *Figure 1*(b), assumes that the slope in the linear elastic behavior is infinite; thus, the elastic motion prior to plasticity onset is zero, and unloading from a plastic state occurs on a vertical line. In practice, the rigid behavior is enforced using an automatically chosen high penalty stiffness.
- Nonlinear elastic behavior, shown in *Figure 1*(c), in which the initial elastic loading occurs along the defined nonlinear path. Elastic unloading occurs along a nonlinear curve ($C \rightarrow O_c$) that is simply the user-defined nonlinear elastic curve motion shifted such that it passes through point C. The user-defined nonlinear elastic behavior must be such that the unloading path ($C \rightarrow O_c$) does not intersect with the loading path ($O \rightarrow I \rightarrow C$); otherwise, a local instability will occur.

Other behaviors (such as damping or friction) can be specified in addition to the elastic/rigid/plastic specifications but will not be considered in the plasticity calculations since they are considered to be in parallel with the elastic-plastic/rigid plastic behavior (see the conceptual model in *Connector Behavior*).

Defining Elastic-Plastic or Rigid Plastic Behavior

As is the case with any other connector behavior type, connector plasticity can be defined only for available components of relative motion. For example, you cannot define plastic behavior in a BEAM connector or in components 2 and 3 of a SLOT connector since these components are not available for behavior definitions. The solution to this problem is to:

- define a connection type with available components of relative motion that best models the kinematics of your connection device both before and after plasticity onset;
- define the desired components as rigid (see *Connector Elastic Behavior*); and
- specify rigid plastic behavior in some or all of these components.

For example, to define rigid plasticity for an otherwise rigid beam-like connector, you could use a PROJECTION CARTESIAN connection together with a PROJECTION FLEXION-TORSION connection, define all components as rigid, and proceed with your plasticity definitions.

Elastic-plastic behavior is usually specified for available components of relative motion for which spring-like behavior is specified and for which plastic deformation may occur.

Input File Usage: Use the following options to define rigid plasticity in connectors:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR ELASTICITY*, RIGID
**CONNECTOR PLASTICITY*
**CONNECTOR HARDENING*

Use the following options to define elastic-plasticity in connectors:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR ELASTICITY*
**CONNECTOR PLASTICITY*
**CONNECTOR HARDENING*

Abaqus/CAE Usage: Use the following input to define rigid plasticity in connectors:

Interaction module: connector section editor: **Add->Elasticity**, **Definition: Rigid**; **Add->Plasticity**

Use the following input to define elastic-plasticity in connectors:

Interaction module: connector section editor: **Add->Elasticity**; **Add->Plasticity**

Defining Uncoupled Plastic Behavior

Uncoupled elastic-plastic or rigid plastic behavior, specified for each component of relative motion independently, is similar to one-dimensional plasticity. You must define elastic or rigid behavior in the specified component of relative motion. In this case the connector potential function is chosen automatically as

$$P(\mathbf{f}) = |f_i|,$$

where f_i is the force or moment in the i^{th} available component of relative motion for which plastic behavior is specified. The associated plastic flow in this case becomes

$$\dot{u}_i^{pl} = \dot{\bar{u}}_i^{pl} \frac{\partial \phi}{\partial f_i} = \dot{\bar{u}}_i^{pl} \, sign(f_i), \quad \text{no sum on } i,$$

where $\dot{u}_i^{pl}$ is the rate of plastic relative motion and $\dot{\bar{u}}_i^{pl}$ is the equivalent plastic relative motion rate in the i^{th} component.

Input File Usage: Use the following options to define uncoupled rigid plastic connector behavior:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR ELASTICITY*, RIGID, COMPONENT=*i*
**CONNECTOR PLASTICITY*, COMPONENT=*i*
**CONNECTOR HARDENING*

Use the following options to define uncoupled elastic-plastic connector behavior:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR ELASTICITY*, COMPONENT=*i*
**CONNECTOR PLASTICITY*, COMPONENT=*i*
**CONNECTOR HARDENING*

Abaqus/CAE Usage: Use the following input to define uncoupled rigid plastic connector behavior:

Interaction module: connector section editor: **Add->Elasticity**, **Definition: Rigid**; **Add->Plasticity**, **Coupling: Uncoupled**

Use the following input to define uncoupled elastic-plastic connector behavior:

Interaction module: connector section editor: **Add->Elasticity**, **Definition: Linear** or **Nonlinear**, **Coupling: Uncoupled**; **Add->Plasticity**, **Coupling: Uncoupled**

Defining Coupled Plastic Behavior

You should define coupled plasticity in connectors when several available components of relative motion are involved simultaneously in a coupled fashion in the definition of the yield function ϕ. In this case you must define the potential, P, via a connector potential definition. Plastic flow eventually occurs only in the intrinsic components of relative motion that are ultimately involved in the potential. Elastic or rigid behavior should be specified for all components of relative motion that are involved in the potential definition. The elastic/rigid behavior for these components can be specified in an uncoupled fashion, in a coupled fashion, or in a combination of both. All elasticity definitions specified in a connector behavior that are pertinent to the components of relative motion involved in the potential definition are used collectively to define the elasticity for the coupled elastic-plastic or rigid plastic definition.

Input File Usage: Use the following options to define coupled elastic-plastic or rigid plastic connector behavior:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR ELASTICITY*
**CONNECTOR PLASTICITY*
**CONNECTOR POTENTIAL*
**CONNECTOR HARDENING*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Elasticity**; **Add->Plasticity**, **Coupling: Coupled**, **Force Potential**

Mode Mix Ratio

If the coupled plasticity definition includes at least two terms in the associated potential definition (see *Defining Derived Components for Connector Elements*), a mode mix ratio can be defined to reflect the relative weight of the first two terms in their contribution to the potential. The mode mix ratio can be used in connector hardening definitions to specify dependencies in equivalent yield force or moment and, in plastic motion-based connector damage definitions (see *Connector Damage Behavior*), to specify dependencies in both damage initiation and damage evolution. It is defined as

$$\Psi_m = \left(\frac{2}{\pi}\right) tan^{-1}\left(\frac{F_I}{F_{II}}\right),$$

where F_I is the force/moment in the first component specified for the plasticity potential and F_{II} is the force/moment in the second component specified for the same potential. $\Psi_m = 0.0$ if $F_I = 0.0$, $\Psi_m = 1.0$ if $F_{II} = 0.0$, and Ψ_m is somewhere in between −1.0 and 1.0 if neither is 0.0.

Defining the Plastic Hardening Behavior

Abaqus provides a number of hardening models varying from simple perfect plasticity to nonlinear isotropic/kinematic hardening. Connector hardening is analogous to the hardening models used in Abaqus for metals subjected to cyclic loading and described in *Models for Metals Subjected to Cyclic Loading*.

Defining Perfect Plasticity

Perfect plasticity means that the yield force does not change with plastic relative motion.

Input File Usage: Use the following option to define perfect plasticity:

**CONNECTOR HARDENING*

$F|_0$

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Plasticity**: **Specify isotropic hardening**, **Isotropic Hardening**, and enter the **Yield Force/Moment** in the data table

Defining Nonlinear Isotropic Hardening

Isotropic hardening behavior defines the evolution of the yield surface size, F^0, as a function of the equivalent plastic relative motion, $\overline{u}^{pl}$. This evolution can be introduced by specifying F^0 directly as a function of $\overline{u}^{pl}$ in tabular form or by using the simple exponential law

$$F^0 = F|_0 + Q_{\text{inf}}\left(1 - e^{-b\overline{u}^{pl}}\right),$$

where $F|_0$ is the yield value at zero plastic relative motion and Q_{inf} and b are material parameters. Q_{inf} is the maximum change in the size of the yield surface, and b defines the rate at which the size of the yield surface changes as plastic deformation develops. When the equivalent force defining the size of the yield surface remains constant ($F^0 = F|_0$), there is no isotropic hardening.

Defining the Isotropic Hardening Component by Specifying Tabular Data

You can introduce isotropic hardening by specifying the equivalent force or moment defining the size of the yield surface, F^0, as a tabular function of the equivalent relative plastic motion, $\bar{u}^{pl}$, and, if required, of the mode mix ratio, Ψ_m (see *Mode Mix Ratio*), the equivalent relative plastic motion rate, $\dot{\bar{u}}^{pl}$, temperature, and other predefined field variables. The yield value at a given state is interpolated from this table of data.

Input File Usage: Use the following option to define connector hardening behavior that is independent of mode mix:

**CONNECTOR HARDENING*, TYPE=ISOTROPIC, DEFINITION=TABULAR (default)

Use the following option to define mode mix dependent connector hardening behavior:

**CONNECTOR HARDENING*, TYPE=ISOTROPIC, DEFINITION=TABULAR (default), MODE MIX DEPENDENT

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Plasticity**: **Specify isotropic hardening**, **Isotropic Hardening**, **Definition: Tabular**

Mode mix dependent connector hardening behavior is not supported in Abaqus/CAE.

Defining the Isotropic Hardening Component Using the Exponential Law

Specify the material parameters of the exponential law ($F|_0$, $Q_{\inf}$, and b) directly if they are already calibrated from test data. These parameters can be specified as functions of temperature and/or field variables.

Input File Usage: **CONNECTOR HARDENING*, TYPE=ISOTROPIC, DEFINITION=EXPONENTIAL LAW

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Plasticity**: **Specify isotropic hardening**, **Isotropic Hardening**, **Definition: Exponential law**

Defining Nonlinear Kinematic Hardening

When nonlinear kinematic hardening is specified, the center of the yield surface is allowed to translate in the force space. The backforce, $\boldsymbol{\alpha}$, is the current center of the yield surface and is interpreted similar to the backstress $\boldsymbol{\alpha}$ discussed in *Classical Metal Plasticity*.

The yield surface is defined by the function

$$\phi := P(\mathbf{f} - \boldsymbol{\alpha}) - F^0 \leq 0,$$

where F^0 is the yield value and $P(\mathbf{f} - \boldsymbol{\alpha})$ is the potential with respect to the backforce $\boldsymbol{\alpha}$.

The kinematic hardening component is defined to be an additive combination of a purely kinematic term (the linear Ziegler hardening law) and a relaxation term (the recall term) that introduces the nonlinearity. When temperature and field variable dependencies are omitted, the hardening law is

$$\dot{\boldsymbol{\alpha}} = C\frac{1}{F^0}(\mathbf{f} - \boldsymbol{\alpha})\dot{\bar{u}}^{pl} - \gamma\boldsymbol{\alpha}\dot{\bar{u}}^{pl},$$

where C and γ are material parameters that must be calibrated from cyclic test data. C is the initial kinematic hardening modulus, and γ determines the rate at which the kinematic hardening modulus decreases with increasing plastic deformation. When C and γ are zero, the model reduces to an isotropic hardening model. When γ is zero, the linear Ziegler hardening law is recovered. Refer to *Models for Metals Subjected to Cyclic Loading* for a discussion of calibrating the material parameters.

Defining the Kinematic Hardening Component by Specifying Half-Cycle Test Data

If limited test data are available, C and γ can be based on the force-constitutive motion data obtained from the first half cycle of a unidirectional tension or compression experiment. An example of such test data is shown in *Figure 2*.

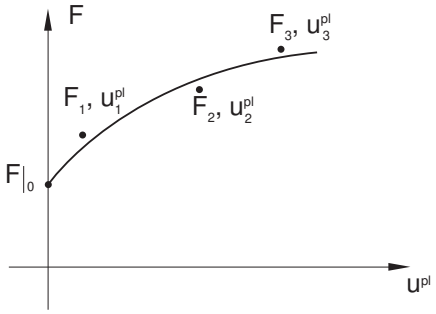


Figure 2: Half-cycle of force-motion data.

This approach is usually adequate when the simulation will involve only a few cycles of loading.

For each data point (F_j, u_j^{pl}) a value of α_j is obtained from the test data as

$$\alpha_j = F_j - F_j^0,$$

where F_j^0 is the user-defined size of the yield surface at the corresponding plastic motion for the isotropic hardening definition or the initial yield force if the isotropic hardening component is not defined.

Integration of the backforce evolution law over a half cycle yields the expression

$$\alpha = \frac{C}{\gamma}\left(1 - e^{-\gamma u^{pl}}\right),$$

which is used for calibrating C and γ.

When test data are given as functions of temperature and/or field variables, it is recommended that a data check analysis be run first. During the data check run, Abaqus will determine several pairs of material parameters (C, γ), where each pair will correspond to a given combination of temperature and/or field variables. Since Abaqus requires the parameter γ to be a constant, the data check analysis will terminate with an error message if γ is not a constant. However, an appropriate constant value of γ may be determined from the information provided in the data file during the data check run. The values for the parameter C and the constant γ can then be entered directly as described below.

Input File Usage: **CONNECTOR HARDENING*, TYPE=KINEMATIC, DEFINITION=HALF CYCLE (default)

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Plasticity**: **Specify kinematic hardening**, **Kinematic Hardening**, **Definition: Half-cycle**

Defining the Kinematic Hardening Component by Specifying Test Data from a Stabilized Cycle

Force-constitutive motion data can be obtained from the stabilized cycle of a specimen that is subjected to symmetric cycles. A stabilized cycle is obtained by cycling the specimen over a fixed motion range Δu until a steady-state condition is reached; that is, until the force-motion curve no longer changes shape from one cycle to the next. Such a stabilized cycle is shown in *Figure 3*.

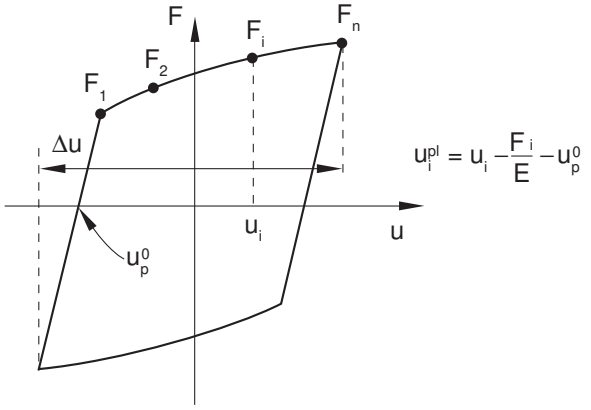


Figure 3: Force-motion data for a stabilized cycle.

See *Models for Metals Subjected to Cyclic Loading* for information on how the data should be processed before they are specified in the connector hardening definition.

Input File Usage: **CONNECTOR HARDENING*, TYPE=KINEMATIC, DEFINITION=STABILIZED

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Plasticity**: **Specify kinematic hardening**, **Kinematic Hardening**, **Definition: Stabilized**

Defining the Kinematic Hardening Component by Specifying the Material Parameters Directly

The parameters C and γ can be specified directly if they are already calibrated from test data. The parameter C can be provided as a function of temperature and/or field variables, but temperature and field variable dependence of γ is not available. The algorithm currently used to integrate the nonlinear isotropic/kinematic hardening model does not provide accurate solutions if the value of γ changes significantly in an increment due to temperature and/or field variable dependence.

Input File Usage: **CONNECTOR HARDENING*, TYPE=KINEMATIC, DEFINITION=PARAMETERS

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Plasticity**: **Specify kinematic hardening**, **Kinematic Hardening**, **Definition: Parameters**

Defining Nonlinear Isotropic/Kinematic Hardening

The evolution law of the combined isotropic/kinematic model consists of two components: an isotropic hardening component, which describes the change in the equivalent force defining the size of the yield surface, F^0, as a function of plastic relative motion, and a nonlinear kinematic hardening component, which describes the translation of the yield surface in force space through the backforce, α.

At most two connector hardening definitions, one isotropic and one kinematic, can be associated with a connector plasticity definition. If only one connector hardening definition is specified, it can be either isotropic or kinematic.

Input File Usage: Use the following two options to define nonlinear isotropic/kinematic hardening:

**CONNECTOR HARDENING*, TYPE=KINEMATIC
**CONNECTOR HARDENING*, TYPE=ISOTROPIC

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Plasticity**: **Specify isotropic hardening** and **Specify kinematic hardening**

Using Multiple Plasticity Definitions

Multiple connector plasticity definitions can be used as part of the same connector behavior definition. However, only one connector plasticity definition can be used to define plasticity for each available component of relative motion. At most one coupled plasticity definition can be associated with a connector behavior definition. Additional connector plasticity definitions are permitted for the same connector behavior definition only if the two spaces do not overlap; for example, you could define uncoupled connector plasticity for components 1, 2, and 6 and have one coupled connector plasticity definition involving components 3, 4, and 5.

Each connector plasticity definition must have its own hardening definition.

Examples

Illustrations of uncoupled and coupled plasticity behaviors are shown in the following examples.

Uncoupled Plasticity in a SLOT-Like Connector

Consider a SLOT connector that you have used to model a physical device efficiently. You have examined the reaction forces enforcing the SLOT constraint in the local 2- and 3-directions; since they appear to be quite large, you need to assess whether plastic deformations in the device may occur. One option that you have is to create detailed meshes for the slot and the pin in the device, define the contact interactions between them, and use elastic-plastic material definitions for the underlying materials. While this is the most accurate modeling solution, it may be impractical, especially when the device you are modeling is part of a larger model. Alternatively, you can do the following:

- use a CARTESIAN connection type instead of the SLOT connection with the first axis aligned with the slot direction;
- define components 2 and 3 as rigid; and
- define rigid plasticity separately in each of the components.

The following input can be used:

```
*CONNECTOR SECTION, BEHAVIOR=slot
CARTESIAN
 orientation at node a
*CONNECTOR BEHAVIOR, NAME=slot
*CONNECTOR ELASTICITY, RIGID
 2, 3
*CONNECTOR PLASTICITY, COMPONENT=2
*CONNECTOR HARDENING, TYPE=ISOTROPIC
 100, 0.0
 110, 0.12
*CONNECTOR PLASTICITY, COMPONENT=3
*CONNECTOR HARDENING, TYPE=ISOTROPIC
 50, 0.0
 75, 0.23
```

The yield forces that you specify in the connector hardening definitions are obtained from an experimental result or are assessed from a “virtual experiment,” as follows:

- Use the meshed model of the slot discussed above.
- Run two simple separate analyses by constraining the slot part of the device and driving the pin into the slot walls using a boundary condition.
- Plot the reaction force at the pin node against its motion.
- Use these data to create the force-motion hardening curve to be specified in the connector hardening definition.

Coupled Plasticity in a Spot Weld

Referring to the spot weld shown in *Figure 4* and to the yield function described in *Defining Connector Potentials*,

$$\phi(\mathbf{f}) = \left[\left(\frac{max(F_n,0)}{R_n}\right)^a + \left(\frac{|F_s|}{R_s}\right)^a\right]^{1/a},$$

you could complete the plasticity definition, for example, by specifying tabular isotropic hardening and kinematic hardening via parameters.

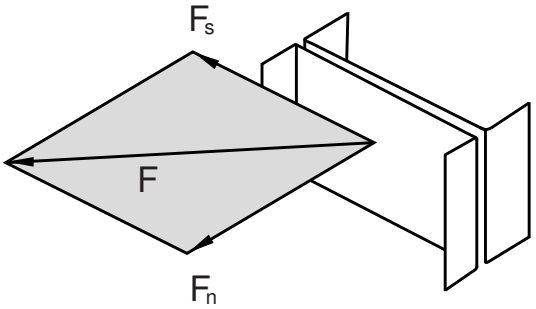


Figure 4: Spot weld connection.

```
*PARAMETER
R_n=0.02
R_s=0.05
*CONNECTOR ELASTICITY, RIGID
*CONNECTOR PLASTICITY
*CONNECTOR POTENTIAL, EXPONENT=a
normal, R_n, , MACAULEY
shear, R_s, , ABS
*CONNECTOR HARDENING, TYPE=ISOTROPIC
F_1^0, ū_1^pl
F_2^0, ū_2^pl
*CONNECTOR HARDENING, TYPE=KINEMATIC, DEFINITION=PARAMETERS
C, γ
```

Defining Plastic Connector Behavior in Linear Perturbation Procedures

Plastic relative motions are not allowed during linear perturbation analyses. Therefore, the connector relative motions will be linear elastic perturbations about the plastically deformed base state, similar to metal plasticity.

Output

The Abaqus output variables available for connectors are listed in *Abaqus/Standard Output Variable Identifiers* and *Abaqus/Explicit Output Variable Identifiers*. The following output variables are of particular interest when defining plasticity in connectors:

CUE
Connector elastic displacements/rotations.

CUP
Connector plastic displacements/rotations.

CUPEQ
Connector equivalent plastic relative displacements/rotations. In addition to the usual six components associated with connector output variables, CUPEQ includes the scalar CUPEQC, which is the equivalent plastic relative motion associated with a coupled plasticity definition.

CALPHAF
Connector kinematic hardening shift forces/moments.

Connector Damage Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Connectors*
- *Connector Behavior*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR DAMAGE EVOLUTION*
- **CONNECTOR DAMAGE INITIATION*
- **CONNECTOR ELASTICITY*
- **CONNECTOR PLASTICITY*
- **CONNECTOR POTENTIAL*
- **SECTION CONTROLS*
- *Defining damage*

Overview

Connector damage behavior:

- can be specified in any connectors with available components of relative motion;
- can be used to degrade the elastic, elastic-plastic, or rigid plastic response in connector elements;
- can use a force-based, motion-based, or plastic motion–based damage initiation criterion upon which response degradation may be triggered;
- can use either a (plastic) motion-based or an energy-based damage evolution law to degrade the force response in the connector;
- can be defined in terms of several competing damage mechanisms; and
- can be used only as an indicator of proximity to the damage initiation point without degrading the connector response.

Damage Formulation in Connectors

If relative forces or motions in a connection exceed critical values, the connector starts undergoing irreversible damage (degradation). Upon additional loading there is further evolution of damage leading to eventual failure. If damage has occurred, the force response in the connector component i will change according to the following general form:

$$F_i = (1 - d_i)\, F_{eff_i} \ , \quad 0 \le d_i \le 1 \qquad \text{no sum on i}$$

where d_i is a scalar damage variable and F_{eff_i} is the response in the available connector component of relative motion i if damage were not present (effective response).

To define a connector damage mechanism, you specify the following:

- a criterion for damage initiation; and
- a damage evolution law that specifies how the damage variable d evolves (optional).

Prior to damage initiation, d has a value of 0.0; thus, the force response in the connector does not change. Once damage has been initiated, the damage variable will monotonically evolve up to the maximum value of 1.0 if damage evolution is specified. Complete failure occurs when $d = 1.0$.

Abaqus allows you to specify a maximum degradation value (the default value is 1.0); damage evolution will stop when the damage variable reaches this value, and the element will be deleted from the mesh by default. Alternatively, you can specify that the damaged connector elements remain in the analysis with no further damage evolution. The maximum degradation value is used to evaluate the damaged stiffness in the remaining part of the analysis. This functionality is discussed in detail in *Controlling Element Deletion and Maximum Degradation for Materials with Damage Evolution*.

For connectors with purely elastic behavior, damage can be initiated and evolved in one direction only. If damage was initiated in tension, it will evolve in tension; if damage was initiated in compression, it will evolve in compression. Once damage initiates in tension, it cannot be initiated in compression and vice versa.

Defining Connector Damage Initiation

The degradation process in connectors initiates when forces or relative motions in the connector satisfy certain criteria. Three different criteria types can be used to trigger damage in connectors: criteria based on force, plastic motion, or constitutive motion. Connector damage initiation criteria for the available components of relative motion can be specified for each component independently (uncoupled). Alternatively, connector damage initiation criteria that couple all or some of the available components of relative motion in the connector can be defined.

The damage initiation criterion can depend on temperature and field variables. See *Input Syntax Rules* for further information about defining data as functions of temperature and field variables.

Force-Based Damage Initiation Criterion

By default, the damage initiation criterion is specified in terms of forces/moments in the connector. Elastic or rigid connector behavior must be defined for the components involved in the initiation. You provide the lower (compression) limit, f_{min}, and the upper (tension) limit, f_{max}, for the force/moment damage initiation values. If the force is outside the range specified by the two limit values, damage is initiated. The output variable CDIF can be used to monitor the proximity to the damage initiation point.

Defining Uncoupled Force-Based Damage Initiation

For an uncoupled force-based damage initiation criterion, the connector force in the specified component is compared to the specified limit values. Damage is initiated when the force in the specified component i, f_i, is for the first time outside the range ($f_i \leq f_{min}$ or $f_i \geq f_{max}$).

Input File Usage: **CONNECTOR DAMAGE INITIATION*, COMPONENT=*component number*, CRITERION=FORCE (default), DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damage**: **Coupling: Uncoupled**, **Initiation criterion: Force**

Defining Coupled Force-Based Damage Initiation

For a coupled force-based damage initiation criterion, a connector potential, $P(\mathbf{f})$, must be specified to define an equivalent force magnitude (scalar). The equivalent force magnitude is compared to the specified limit values

to assess damage initiation. Damage is initiated when the equivalent force magnitude, $P(\mathbf{f})$, is for the first time outside the range ($P(\mathbf{f}) \leq f_{min}$ or $P(\mathbf{f}) \geq f_{max}$).

Input File Usage: Use the following options:

**CONNECTOR DAMAGE INITIATION*, CRITERION=FORCE (default), DEPENDENCIES=*n*
**CONNECTOR POTENTIAL*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damage**: **Coupling: Coupled**, **Initiation criterion: Force**, **Initiation Potential**

Plastic Motion–Based Damage Initiation Criterion

The damage initiation criterion can be specified in terms of an equivalent relative plastic motion in the connector. You provide the relative equivalent plastic displacement/rotation at which damage will be initiated as a function of the relative equivalent plastic rate. The output variable CDIP can be used to monitor the proximity to the damage initiation point.

Defining Uncoupled Plastic Damage Initiation

For an uncoupled elastic-plastic or rigid plastic damage initiation criterion, uncoupled connector plasticity in the specified component of relative motion must be defined (see *Connector Plastic Behavior*). When the equivalent relative plastic motion as defined by the associated plasticity definition is greater than the specified limit value for the first time, damage is initiated.

Input File Usage: Use the following options:

**CONNECTOR DAMAGE INITIATION*, COMPONENT=*component number*, CRITERION=PLASTIC MOTION, DEPENDENCIES=*n*
**CONNECTOR PLASTICITY*, COMPONENT=*component number*
or
**CONNECTOR PLASTICITY*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damage**: **Initiation criterion: Plastic motion**; **Add->Plasticity**

Defining Coupled Plastic Damage Initiation

For a coupled elastic-plastic or rigid plastic damage initiation criterion, coupled connector plasticity must be defined. The connector potential used in the coupled connector plasticity function defines an equivalent relative plastic motion. This equivalent relative plastic motion is compared to the specified limit values to assess damage initiation. The equivalent relative plastic motion at which damage is initiated can be a function of the mode mix ratio Ψ_m (see *Connector Plastic Behavior*).

Input File Usage: Use the following options:

**CONNECTOR DAMAGE INITIATION*,
CRITERION=PLASTIC MOTION, DEPENDENCIES=*n*
**CONNECTOR PLASTICITY*
**CONNECTOR POTENTIAL*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damage**: **Coupling: Coupled**, **Initiation criterion: Plastic motion**; **Add->Plasticity**: **Coupling: Coupled**, **Force Potential**

Constitutive Motion-Based Damage Initiation Criterion

The damage initiation criterion can be specified in terms of relative constitutive displacements/rotations in the connector. You provide the lower (compression) limit, u_{min}, and the upper (tension) limit, u_{max}, for the constitutive displacement/rotation damage initiation values. If the motion is outside the range specified by the two limit values, damage is initiated. The output variable CDIM can be used to monitor the proximity to the damage initiation point.

Defining Uncoupled Constitutive Motion-Based Damage Initiation

For an uncoupled motion-based damage initiation criterion, the connector relative constitutive motion in the specified component is compared to the specified limit values. Damage is initiated when the relative constitutive displacement/rotation in the specified component *i*, u_i, is for the first time outside the range ($u_i \le u_{min}$ or $u_i \ge u_{max}$).

Input File Usage: **CONNECTOR DAMAGE INITIATION*, COMPONENT=*component number*, CRITERION=MOTION, DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damage**: **Coupling: Uncoupled**, **Initiation criterion: Motion**

Defining Coupled Constitutive Motion-Based Damage Initiation

For a coupled motion-based damage initiation criterion, a connector potential, $P(\mathbf{u})$, must be specified to define an equivalent motion magnitude (scalar), where $\mathbf{u}$ is the collection of all available components of relative motion in the connector. The equivalent motion magnitude is compared to the specified limit values to assess damage initiation. Damage is initiated when the equivalent motion magnitude, $P(\mathbf{u})$, is for the first time outside the range ($P(\mathbf{u}) \le u_{min}$ or $P(\mathbf{u}) \ge u_{max}$).

Input File Usage: Use the following options:

**CONNECTOR DAMAGE INITIATION*, CRITERION=MOTION, DEPENDENCIES=*n*
**CONNECTOR POTENTIAL*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damage**: **Coupling: Coupled**, **Initiation criterion: Motion**, **Initiation Potential**

Defining Connector Damage Evolution

Connector damage evolution specifies the evolution law for the damage variable. Upon evolution, the connector response will be degraded. The evolution of damage can be based on an energy dissipation criterion or on relative (plastic) motions. In the motion-based criteria the damage variable, *d*, can be defined as a linear, exponential, or tabular function of relative motions.

The damage evolution law can depend on temperature and field variables. See *Input Syntax Rules* for further information about defining data as functions of temperature and field variables.

Specifying the Affected Components

By default (i.e., the affected components are not specified explicitly), only the elastic/rigid or elastic/rigid-plastic response in the connector will be damaged. The response due to friction, damping, and stop/lock behavior will not be degraded. For an uncoupled connector damage mechanism (uncoupled damage initiation criterion), only the specified component of relative motion will undergo damage. For coupled connector damage initiation, the components that will be degraded by default are chosen as follows:

- If a force-based or constitutive motion-based damage initiation criterion is used, the intrinsic available components (1 through 6) that ultimately contribute to the connector potential for damage initiation will be affected.
- If a plastic motion–based damage initiation criterion is used, the intrinsic available components that ultimately contribute to the connector potential used in the coupled plasticity definition will be affected.

Alternatively, you can specify the available components of relative motion that will be affected by the damage evolution law directly. In this case the entire connector response (elasto/rigid-plastic, friction, damping, constraint forces and moments, etc.) in the affected components will be damaged.

Input File Usage: **CONNECTOR DAMAGE EVOLUTION*, AFFECTED COMPONENTS

The first data line identifies the component numbers that will be damaged, and the additional data for the connector damage evolution definition begins on the second data line.

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damage**: **Specify damage evolution**, **Evolution**, **Specify affected components**

Defining a Motion-Based Linear Damage Evolution Law

The linear form of the damage evolution law is illustrated here in the context of linear elasticity, although it can be used in any situation. Assume that the connector response is linear elastic and that after damage initiation a linear damage evolution is desired, as illustrated in *Figure 1*.

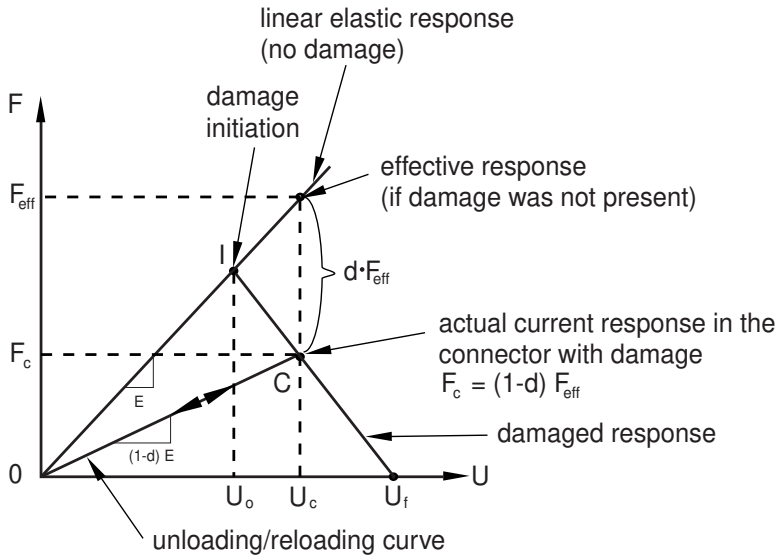


Figure 1: Linear damage evolution law for linear elastic connector behavior.

If damage were not specified, the response would be linear elastic (a straight line passing through the origin). Assume that damage has initiated at point I as triggered by a force-based or motion-based criterion, for example; the corresponding constitutive motion at this point is $\boldsymbol{u_0}$. If the connector is loaded further such that the constitutive motion increases to $\boldsymbol{u_c}$, the connector force response at point C becomes $\boldsymbol{F_c}$. The response is diminished by $\boldsymbol{d * F_{eff}}$ when compared to the effective response $\boldsymbol{F_{eff}}$ (the elastic response with no damage). Thus, $\boldsymbol{F_c = (1-d)\, F_{eff}}$. If unloading occurs at point C, the unloading curve of slope $\boldsymbol{(1-d)\, E}$ is followed. As long as the constitutive motion does not exceed $\boldsymbol{u_c}$, the damage variable, d, stays constant at the value obtained when point C is first reached. If further loading occurs, further damage occurs until the ultimate failure motion, $\boldsymbol{u_f}$, is reached ($d = 1$) and the connector component loses the ability to carry any load. Thus, one possible loading/unloading sequence is O→I→C→O→C → $\boldsymbol{u_f}$.

The linear damage evolution law defines a truly linear damaged force response only in the case of linear elastic or rigid behavior with optional perfect plasticity. If nonlinear elasticity or plasticity with hardening are defined for the damaged components, an approximate linear damaged response is observed.

In Abaqus/Standard the damage force (instead of the damage variable) decays linearly when component-based linear elasticity is defined with component-based damage. This behavior is different from the behavior in Abaqus/Explicit, where the damage variable decays linearly.

Defining the Linear Evolution Law for a Force-Based or Constitutive Motion-Based Damage Initiation Criterion

If an uncoupled damage initiation criterion is used in component i, you specify the difference between the constitutive relative motion at ultimate failure, $\boldsymbol{u_{fi}}$, and the constitutive relative motion at damage initiation, $\boldsymbol{u_{0i}}$, in the specified component ($\boldsymbol{u_{fi} - u_{0i}}$).

If a coupled damage initiation criterion is used, an equivalent constitutive relative motion, $\boldsymbol{\overline{u}}$, must be defined for damage evolution purposes. A connector potential definition is used to define $\boldsymbol{\overline{u} = P(\mathbf{u})}$. You specify the difference between the equivalent motion at ultimate failure, $\boldsymbol{\overline{u}_f}$, and the equivalent motion at damage initiation, $\boldsymbol{\overline{u}_0}$ ($\boldsymbol{\overline{u}_f - \overline{u}_0}$).

Input File Usage: Use the following options to define a linear evolution law for an uncoupled initiation criterion:

**CONNECTOR DAMAGE INITIATION*,
COMPONENT=*component number*, CRITERION=FORCE or MOTION
**CONNECTOR DAMAGE EVOLUTION*, TYPE=MOTION, SOFTENING=LINEAR

Use the following options to define a linear evolution law for a coupled initiation criterion:

**CONNECTOR DAMAGE INITIATION*,
CRITERION=FORCE or MOTION
**CONNECTOR POTENTIAL*
**CONNECTOR DAMAGE EVOLUTION*, TYPE=MOTION,
SOFTENING=LINEAR
**CONNECTOR POTENTIAL*

The second **CONNECTOR POTENTIAL* option defines $\overline{u} = P(\mathbf{u})$.

Abaqus/CAE Usage: Use the following input to define a linear evolution law for an uncoupled initiation criterion:

Interaction module: connector section editor: **Add->Damage**: **Coupling: Uncoupled**, **Initiation criterion: Force** or **Motion**; **Specify damage evolution**, **Evolution type: Motion**, **Evolution softening: Linear**

Use the following input to define a linear evolution law for a coupled initiation criterion:

Interaction module: connector section editor: **Add->Damage**: **Coupling: Coupled**, **Initiation criterion: Force** or **Motion**; **Specify damage evolution**, **Evolution type: Motion**, **Evolution softening: Linear**; **Initiation Potential**; **Evolution Potential**

Defining the Linear Evolution Law for a Plastic Motion–Based Damage Initiation Criterion

You can specify the difference between the associated equivalent plastic relative motion at ultimate failure, $\overline{u}_f^{pl}$, and the associated equivalent plastic relative motion at damage initiation, $\overline{u}_0^{pl}$ ($\overline{u}_f^{pl} - \overline{u}_0^{pl}$), as a function of the mode mix ratio, Ψ_m, defined in *Connector Plastic Behavior*. The equivalent plastic relative motions are calculated from the associated plasticity definition (either coupled or uncoupled).

Input File Usage: Use the following options:

**CONNECTOR DAMAGE INITIATION*, CRITERION=PLASTIC MOTION
**CONNECTOR DAMAGE EVOLUTION*, TYPE=MOTION,
SOFTENING=LINEAR

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damage**: **Initiation criterion: Plastic motion**; **Specify damage evolution**, **Evolution type: Motion**, **Evolution softening: Linear**

Defining a Motion-Based Exponential Damage Evolution Law

The exponential damage evolution law is illustrated in the context of a linear elastic-plastic response with hardening, although it can be used in any situation. The force response in a particular connector component is shown in *Figure 2*.

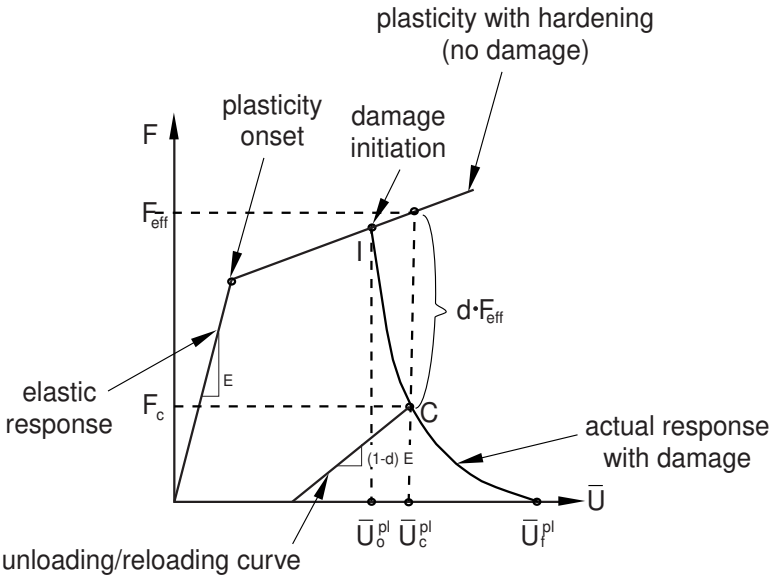


Figure 2: Exponential damage evolution law for linear elastic-plastic connector behavior with hardening.

Assume that damage is initiated at point I as triggered by a plastic motion–based damage initiation criterion. If further loading occurs until point C, the response is $\boldsymbol{F_C = (1-d)\,F_{eff}}$. Unloading from point C occurs along the damaged elastic line of slope $\boldsymbol{(1-d)\,E}$. Upon unloading/reloading, the damage variable remains constant until point C is reached again. Further loading (beyond point C) leads to an increasingly damaged response until the ultimate failure point, $\boldsymbol{\bar{u}_f^{pl}}$, is reached ($d = 1$). The damage variable d is given by the following equation:

$$\boldsymbol{d = \frac{1-e^{-\alpha\frac{\bar{u}^{pl}-\bar{u}_0^{pl}}{\bar{u}_f^{pl}-\bar{u}_0^{pl}}}}{1-e^{-\alpha}}}.$$

The damaged response will appear to be truly exponential only if either linear elasticity or perfect plasticity is used. An approximate exponential degradation is obtained if plasticity with hardening is present.

You specify the difference between the relative motions at ultimate failure and at damage initiation and the exponential coefficient $\boldsymbol{\alpha}$. The difference between the relative motions is interpreted in the same way as described in *Defining a Motion-Based Linear Damage Evolution Law*, as follows:

- If an uncoupled force-based or constitutive motion-based damage initiation criterion is used, the difference between the relative motions at ultimate failure and at damage initiation in the specified component i, $\boldsymbol{u_{fi} - u_{0i}}$, is specified.
- If a coupled force-based or constitutive motion-based damage initiation criterion is used, an equivalent relative motion is defined using a connector potential ($\boldsymbol{\bar{u} = P(\mathbf{u})}$). The difference between the relative motions at ultimate failure and at damage initiation, $\boldsymbol{\bar{u}_f - \bar{u}_0}$, is specified.

- If a plastic motion–based damage initiation criterion is used, the difference between the equivalent relative plastic motions at ultimate failure and at damage initiation, $\overline{u}_f^{pl} - \overline{u}_0^{pl}$, is specified. The equivalent plastic relative motion is calculated from the associated plasticity definition. The data can also be functions of the mode mix ratio Ψ_m.

In the first two cases the equation for the damage variable is similar to that given above for plastic motion–based damage initiation except that (equivalent) constitutive relative motions are used instead of equivalent relative plastic motions.

Input File Usage: **CONNECTOR DAMAGE EVOLUTION*, TYPE=MOTION, SOFTENING=EXPONENTIAL

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damage**: **Specify damage evolution**, **Evolution type: Motion**, **Evolution softening: Exponential**

Defining a Motion-Based Tabular Damage Evolution Law

You can also specify the damage variable directly as a tabular function of the differences between the relative motions at ultimate failure and the relative motions at damage initiation. The differences between the relative motions are interpreted in the same way as described in *Defining a Motion-Based Linear Damage Evolution Law*, as follows:

- If an uncoupled force-based or constitutive motion-based damage initiation criterion is used, the differences between the constitutive relative motions at ultimate failure and at damage initiation in the specified component i, $u_i - u_{0i}$, are used to define the tabular data.
- If a coupled force-based or constitutive motion-based damage initiation criterion is used, an equivalent relative motion is defined using a connector potential ($\overline{u} = P(\mathbf{u})$). The differences between the relative motions at ultimate failure and at damage initiation, $\overline{u} - \overline{u}_0$, are used to define the tabular data.
- If a plastic motion–based damage initiation criterion is used, the differences between the equivalent relative plastic motions at ultimate failure and at damage initiation, $\overline{u}^{pl} - \overline{u}_0^{pl}$, are used. The equivalent plastic relative motion is calculated from the associated plasticity definition. The tabular data can also be functions of the mode mix ratio Ψ_m.

Input File Usage: **CONNECTOR DAMAGE EVOLUTION*, TYPE=MOTION, SOFTENING=TABULAR, DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damage**: **Specify damage evolution**, **Evolution type: Motion**, **Evolution softening: Tabular**

Defining a Damage Evolution Law Using Post-Damage Initiation Dissipation Energies

This damage evolution law is illustrated in the context of nonlinear elasticity, as shown in *Figure 3*.

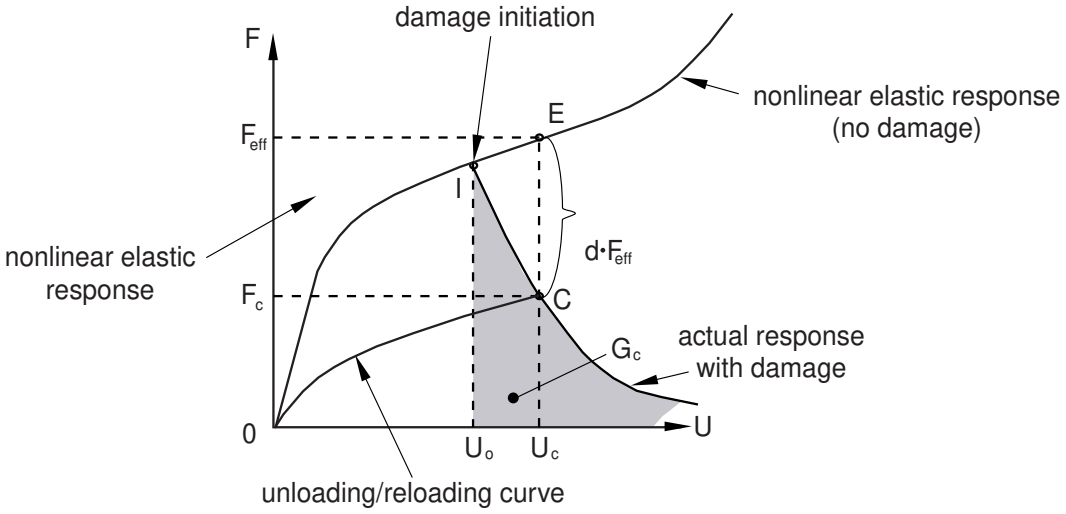


Figure 3: Post-damage initiation dissipation energy evolution law for nonlinear elastic connector behavior.

Assume that damage is initiated at point I when the constitutive relative motion is u_0 as triggered by a force-based or a motion-based damage initiation criterion, for example. The response at point C will be $F_C = (1 - d) F_{eff}$. Unloading from point C occurs along the CO curve, which is the original nonlinear elastic response curve (OE) scaled down by the $(1 - d)$ factor. Damage remains constant on the unloading/reloading curve (C→O→C), and it increases only if loading increases beyond point C.

Instantaneous failure can be specified upon initiation if G_C is specified as 0.0. In all other cases ultimate failure (d = 1) would occur (in theory) at infinite motion since an exponential-like response that asymptotically goes to zero is generated. Abaqus will set d = 1 when the damage dissipated energy reaches $0.99G_C$.

You specify the post-damage initiation dissipated energy at ultimate failure, G_C. If a plastic motion–based initiation criterion is used, G_C can be specified as a function of the mode mix ratio Ψ_m.

Input File Usage: **CONNECTOR DAMAGE EVOLUTION*, TYPE=ENERGY, DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damage**: **Specify damage evolution**, **Evolution type: Energy**

Using Multiple Damage Mechanisms

At most three uncoupled damage mechanisms (pairs of connector damage initiation criteria and connector damage evolution laws) can be defined for each available component of relative motion, one for each type of initiation criterion (force, motion, and plastic motion). In addition, three coupled damage mechanisms can be defined (one for each type of initiation criterion). Coupled and uncoupled damage definitions can be combined; only one overall damage variable per component will be used to damage the response in a particular available component of relative motion. Only the overall damage will be output.

Specifying the Contribution of Each Damage Mechanism

When several damage mechanisms are defined for the same connector behavior, you can specify the contribution of each damage mechanism to the overall damage effect for a particular component of relative motion. By default, the damage value associated with a particular mechanism will be compared to the damage values from any other

damage mechanisms defined for the connector behavior, and only the maximum value will be considered for the overall damage. Alternatively, you can specify that the damage values for the mechanisms associated with the connector behavior should be combined in a multiplicative fashion to obtain the overall damage. See the last example below for an illustration.

Input File Usage: Use the following option to specify that only the maximum damage value associated with a particular connector behavior should contribute to the overall damage effect:

**CONNECTOR DAMAGE EVOLUTION*, DEGRADATION=MAXIMUM

Use the following option to specify that all the damage values associated with a particular connector behavior should contribute in a multiplicative way to the overall damage effect:

**CONNECTOR DAMAGE EVOLUTION*, DEGRADATION=MULTIPLICATIVE

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Damage**: **Specify damage evolution**, **Evolution**, **Degradation: Maximum** or **Multiplicative**

Examples

The examples that follow illustrate several methods for defining damage mechanisms.

Uncoupled Damage

The following input could be used to define a simple uncoupled damage mechanism:

```
*CONNECTOR ELASTICITY, COMPONENT=1
*CONNECTOR DAMAGE INITIATION, COMPONENT=1, CRITERION=FORCE
force_compress, force_tens
*CONNECTOR DAMAGE EVOLUTION, TYPE=ENERGY
0.0
```

Damage will initiate when the elastic force in component 1 is either smaller than *force_compress* or larger than *force_tens*. Only the elastic response in component 1 will be damaged. Since the dissipated energy specified for damage evolution is 0.0, the damage evolves catastrophically instantaneously after it has initiated.

Coupled Rigid Plasticity with Plasticity-Based Damage

Referring to the spot weld in *Figure 4* for which coupled plasticity is defined in *Connector Plastic Behavior*, a plastic motion–based damage initiation and evolution with dependencies on the mode mix ratio can be specified as follows:

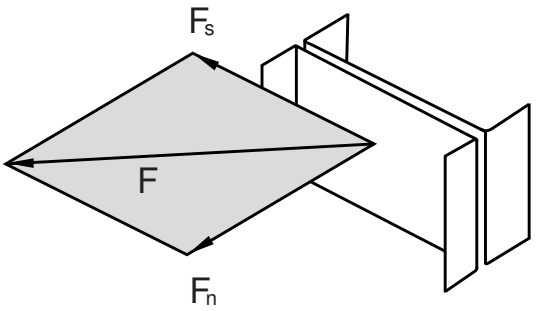


Figure 4: Spot weld connection.

```
*PARAMETER
```
$\overline{u}^{pl}_{init_{0.0}}$=0.25

$\overline{u}^{pl}_{init_{0.5}}$=0.35

$\overline{u}^{pl}_{init_{1.0}}$=0.45

$\overline{u}^{pl}_{evol_{0.0}}$=0.75

$\overline{u}^{pl}_{evol_{0.3}}$=0.78

$\overline{u}^{pl}_{evol_{0.5}}$=0.82

$\overline{u}^{pl}_{evol_{1.0}}$=0.85
```
*CONNECTOR DAMAGE INITIATION, CRITERION=PLASTIC MOTION
```
$< \overline{u}^{pl}_{init_{0.0}} >$, 0.0

$< \overline{u}^{pl}_{init_{0.5}} >$, 0.5

$< \overline{u}^{pl}_{init_{1.0}} >$, 1.0
```
*CONNECTOR DAMAGE EVOLUTION, TYPE=MOTION, SOFTENING=LINEAR
```
$< \overline{u}^{pl}_{evol_{0.0}} >$, 0.0

$< \overline{u}^{pl}_{evol_{0.3}} >$, 0.3

$< \overline{u}^{pl}_{evol_{0.5}} >$, 0.5

$< \overline{u}^{pl}_{evol_{1.0}} >$, 1.0

The equivalent plastic relative motion on the data lines is defined by the associated coupled plasticity definition illustrated in *Connector Plastic Behavior*. For the damage evolution the post-damage-initiation equivalent plastic relative motion should be specified. The second column in all the data lines represents the mode mix ratios as defined in *Connector Plastic Behavior*. In this particular case the mode mix ratio is $\left(\frac{2}{\pi}\right) tan^{-1}\left(F_n/F_s\right)$. The data point at 0.0 comes from a pure “shear” experiment, and the data point at 1.0 comes from a pure “normal” experiment. Data for the values in between come from combined “shear-normal” experiments.

Coupled Rigid Plasticity with Force-Based Damage Initiation and Motion-Based Damage Evolution

Referring to the spot weld in *Figure 4* and using the derived components `normal` and `shear` defined in *Defining Derived Components for Connector Elements*, an alternative way to define damage in the spot weld is to use:

```
*PARAMETER
```
$exponent$=2

```
u_fail^post=0.85
R_n=120.0
R_n=115.0
*CONNECTOR DAMAGE INITIATION, CRITERION=FORCE
, 1.0
*CONNECTOR POTENTIAL
normal, < R_n >
shear, < R_s >
** sqrt((F_n/R_n)^2 + (F_s/R_s)^2)
*CONNECTOR DAMAGE EVOLUTION, TYPE=MOTION, SOFTENING=EXPONENTIAL
< u_fail^post >, < exponent >
*CONNECTOR POTENTIAL
1
2
3
** ū = sqrt(u_1^2 + u_2^2 + u_3^2)
```

Damage will be initiated when the force magnitude defined by the first connector potential definition exceeds the specified value of 1.0. The scale factors R_n and R_s in the first potential definition are used in this case to define a force magnitude that would be 1.0 at damage initiation. A motion-based exponential decay damage evolution law is chosen. The second connector potential definition is associated with the connector damage evolution definition and defines an equivalent motion, $\bar{u}$, in the connection. When the equivalent post-initiation motion, $\bar{u} - \bar{u}_0$ (where $\bar{u}_0$ is $\bar{u}$ at damage initiation), reaches u_{fail}^{post}, ultimate failure occurs. All components (1 through 6) are affected in this case since they all ultimately contribute to the first connector potential definition (see *Defining Derived Components for Connector Elements* for the specific definitions associated with the `normal` and `shear` derived components).

Elastic-Plasticity with Four Competing Damage Mechanisms

This example illustrates how to specify the contributions of multiple damage mechanisms to the overall damage effect and the components of relative motion affected by the damage evolution law. Most of the data line entries or parameters are not given for conciseness.

```
** first damage mechanism: force-based damage initiation
** damage variable d^F
*CONNECTOR DAMAGE INITIATION, COMPONENT=4, CRITERION=FORCE
*CONNECTOR DAMAGE EVOLUTION, TYPE=MOTION, SOFTENING=EXPONENTIAL,
DEGRADATION=MAXIMUM, AFFECTED COMPONENTS
4, 6
**
** second damage mechanism: motion-based damage initiation
** damage variable d^M
*CONNECTOR DAMAGE INITIATION, COMPONENT=4, CRITERION=MOTION
*CONNECTOR DAMAGE EVOLUTION, TYPE=MOTION, SOFTENING=LINEAR,
DEGRADATION=MULTIPLICATIVE, AFFECTED COMPONENTS
1, 2, 6
**
** third damage mechanism: plastic motion-based damage initiation
** damage variable d^P
*CONNECTOR DAMAGE INITIATION, COMPONENT=4,
```

```
CRITERION=PLASTIC MOTION
*CONNECTOR DAMAGE EVOLUTION, TYPE=MOTION, SOFTENING=TABULAR,
DEGRADATION=MULTIPLICATIVE, AFFECTED COMPONENTS
1, 2
**
** fourth damage mechanism: coupled force-based damage initiation
** damage variable d^CF
*CONNECTOR DAMAGE INITIATION, CRITERION=FORCE
*CONNECTOR POTENTIAL
** using components 1, 2, 3, 4, 5, 6
*CONNECTOR DAMAGE EVOLUTION, TYPE=ENERGY, DEGRADATION=MAXIMUM,
AFFECTED COMPONENTS
1, 3, 4, 6
```

Four damage mechanisms (connector damage initiation/connector damage evolution pairs) are specified: three uncoupled and one coupled. The first line of each damage evolution definition establishes the components that will be damaged by the mechanism. The overall damage in a particular component is determined by contributions from all the mechanisms that affect that component. For example, the overall damage in component 1, d_1, is determined by the second, third, and fourth damage mechanisms as follows:

$$1-d_1 = min\left[\left(1-d^M\right) * \left(1-d^P\right), \left(1-d^{CF}\right)\right].$$

d^M and d^P use multiplicative degradation; therefore, they are multiplied first: $\left(1-d^M\right) * \left(1-d^P\right)$. d^{CF} uses maximum degradation, so $\left(1-d^{CF}\right)$ is compared to $\left(1-d^M\right) * \left(1-d^P\right)$ and the minimum value is taken.

For example, assume that at a particular time t, d^M=0.5, d^P=0.3, and d^{CF}=0.2 and at time $t+\Delta t$, d^M=0.6 (the only one increasing) while d^P and d^{CF} stay the same. The overall damage variable gets closer to the ultimate damage value faster when all three damage mechanisms are used than if we use only the d^M mechanism:

$$1-d_{1|_t} = min\left[(1-0.5) * (1-0.3), (1-0.2)\right] = 0.15,$$

while

$$1-d_{1|_{t+\Delta t}} = min\left[(1-0.6) * (1-0.3), (1-0.2)\right] = 0.12.$$

Complete failure occurs when $1-d_1$ reaches 0.0.

$F_i = (1-d_i) * F_{eff_i}$, where i refers to the i^{th} available component of relative motion. The overall damage variables for the other components are determined as follows (based on the specified affected components for each damage evolution law):

$$\begin{aligned}
1-d_2 &= \left(1-d^M\right) * \left(1-d^P\right) \\
1-d_3 &= \left(1-d^{CF}\right) \\
1-d_4 &= min\left[\left(1-d^F\right), \left(1-d^{CF}\right)\right] \\
1-d_5 &= 1 \quad (\text{not damaged}) \\
1-d_6 &= min\left[\left(1-d^F\right) * \left(1-d^M\right), \left(1-d^{CF}\right)\right]
\end{aligned}$$

Maximum Degradation and Choice of Element Removal in Abaqus/Standard

You have control over how Abaqus/Standard treats connector elements with severe damage. By default, the upper bound to the overall damage variable at a material point is $D_{max} = 1.0$. You can reduce this upper bound as discussed in *Controlling Element Deletion and Maximum Degradation for Materials with Damage Evolution*.

By default, once the overall damage variable in at least one component reaches D_{max}, the connector elements are removed (deleted). See *Controlling Element Deletion and Maximum Degradation for Materials with Damage Evolution* for details. Once removed, connector elements offer no resistance to subsequent deformation.

Alternatively, you can specify that a connector element should remain in the model even after the overall damage variable reaches D_{max}. In this case, once the overall damage variable reaches D_{max}, the element stiffness remains constant at $(1 - D_{max})$ times the undamaged stiffness.

Viscous Regularization in Abaqus/Standard

Damage causes a softening response in connector elements, which often leads to convergence difficulties in an implicit code such as Abaqus/Standard. One technique for overcoming convergence difficulties is applying viscous regularization to the constitutive response by introducing a viscous damage variable, d_i^v, as defined by the evolution equation

$$\dot{d}_i^v = \frac{1}{\mu}\left(d_i - d_i^v\right),$$

where d_i is the damage variable evaluated in the inviscid backbone model and μ is the viscosity parameter representing the relaxation time. The damaged response of the viscous material is given as

$$F_i = \left(1 - d_i^v\right) F_{eff_i}.$$

As a result of viscous regularization, the damped damage variable does not obey the specified evolution law exactly (only the backbone damage variable does).

Input File Usage: **SECTION CONTROLS*, NAME=*name*, VISCOSITY=μ
**CONNECTOR SECTION*, CONTROLS=*name*

Abaqus/CAE Usage: Viscous regularization is not supported in Abaqus/CAE.

Defining Connector Damage Behavior in Linear Perturbation Procedures

Damage cannot be initiated and damage variables do not evolve during linear perturbation analyses. Consequently, during a linear perturbation step damage is "frozen" in the state attained at the end of the previous general step.

Output

The Abaqus output variables available for connectors are listed in *Abaqus/Standard Output Variable Identifiers* and *Abaqus/Explicit Output Variable Identifiers*. The following variables are of particular interest when damage is defined in connectors:

CDMG Connector overall damage variable.

CDIF Force-based connector damage initiation variable. In addition to the usual six components associated with connector output variables, CDIF includes the scalar CDIFC, which is the damage initiation criterion value associated with a coupled force-based damage initiation criterion.

CDIM Motion-based connector damage initiation variable. CDIM includes the scalar CDIMC, which is the damage initiation criterion value associated with a coupled motion-based damage initiation criterion.

CDIP Plastic motion–based connector damage initiation variable. CDIP includes the scalar CDIPC, which is the damage initiation criterion value associated with a coupled plastic motion based damage initiation criterion.

ALLDMD Energy dissipated by damage.

ALLCD Energy dissipated by viscous regularization.

Connector Stops and Locks

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Connectors*
- *Connector Behavior*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR LOCK*
- **CONNECTOR STOP*
- *Defining a stop*
- *Defining a lock*

Overview

Connector stops and locks can be:

- specified in any connector with available components of relative motion;
- used to specify contact-enforced stops in individual components of relative motion; and
- used to lock in position an available component of relative motion when a certain criterion is met.

Defining Connector Stops

In the physical construction of most connectors, the admissible position of one body relative to the other is limited by a certain range. In Abaqus these limits are modeled as built-in inequality constraints. You specify the available components of relative motion for which the connector stops are to be defined and the lower and upper limit values of the connector's admissible range of positions in the directions of the components of relative motion.

Input File Usage: Use the following options to define a connector stop:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR STOP*, COMPONENT=*component number*
lower limit, *upper limit*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Stop**: **Components:** *component or components*, **Lower bound:** *lower limit*, **Upper bound:** *upper limit*

Example

Since the shock in *Figure 1* has finite length, contact with the ends of the shock determines the upper and lower limit values of the distance that node *b* can be from node *a*.

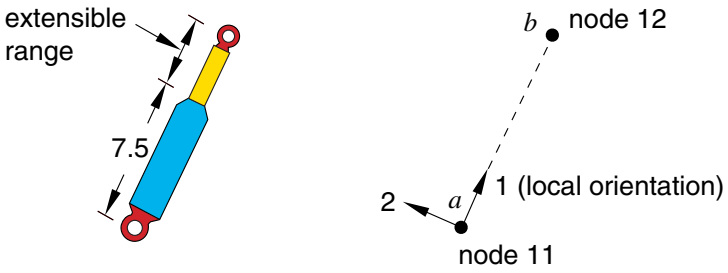


Figure 1: Simplified connector model of a shock absorber.

Assume that the maximum length of the shock is 15.0 units and that the minimum length is 7.5 units. Modify the input file presented in *About Connectors* that is associated with the example in *Figure 1* to include the following lines:

```
*CONNECTOR BEHAVIOR, NAME=sbehavior
...
*CONNECTOR STOP, COMPONENT=1
7.5, 15.0
```

Defining Connector Locks

Connector mechanisms might have devices designed to lock the connector in place once a desired configuration is achieved. For example, a revolute connection might have a falling-pin mechanism that locks the rotational motion after achieving a desired angle. A user-defined connector locking criterion can be defined for connector elements that contain available components of relative motion. You can select the component of relative motion for which the locking criterion is defined.

Connector locks can be used to specify connector behavior for constrained as well as available components of relative motion. Limit values for force or moment can be specified for all components of relative motion involved in the connection. The force/moment used in evaluating the criterion is as computed in the output variable CTF. In addition, limit values can be specified for relative position corresponding to the available components of relative motion. If no other behavior is specified for an available component of relative motion, a force locking criterion will not be useful because CTF is zero.

In Abaqus/Explicit you can also specify the limiting values of velocity in the available components as a criterion for locking. Velocity-dependent locking criteria are useful in modeling seatbelt systems in automobiles (see *Seat belt analysis of a simplified crash dummy*). Moreover, the limiting values can be dependent on temperature and field variables. Field variable dependencies can be used to model time-dependent locks.

If the locking criterion specified for the selected component of relative motion is met, either all components lock or a single available component locks in place. By default, all components of relative motion are locked in place upon meeting the locking criterion. In this case the connector element will be completely kinematically locked from that point on. In dynamic analyses this locking may introduce high accelerations. You can specify if only a selected component of relative motion is locked.

Input File Usage: Use the following options to define a connector lock:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR LOCK*, COMPONENT=*component number*,
LOCK=ALL or *component number*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Lock**: **Components:** *component or components*, **Lock: All** or **Specify** *component*

Example

In the example in *Figure 1* assume that relative rotations about the shock will lock if the force in the local 3-direction exceeds 500.0 units of force.

```
*CONNECTOR BEHAVIOR, NAME=sbehavior
*CONNECTOR LOCK, COMPONENT=3, LOCK=4
, , -500.0, 500.0
```

Defining Connector Stops and Locks in Linear Perturbation Procedures

The status of connector locks or stops cannot change during a linear perturbation analysis; all connector stop and connector lock definitions remain in the same status as in the base state.

Limitations

Activating a connector stop in a large-displacement analysis can require relatively small time increments so that the deformation history is continuous in the vicinity of the value of the relative motion at which you activated the stop. Large time increments can lead to a discontinuous deformation response that violates the connector stop.

An example of such a behavior is illustrated in *Figure 2* in the context of a simple axial connector that is fixed at one end (node 1) and has an applied compressive load at the other end (node 2). Assume a specified connector stop behavior that is activated when the distance between the two nodes is half its initial value. If the compressive load is large enough and applied in a single increment, the displacement of node 2 can be so large that it ends up on the opposite side of node 1 (as shown in *Figure 2*), thereby violating the connector stop. Using smaller time increments would lead to a more continuous response that respects the specified connector stop behavior.

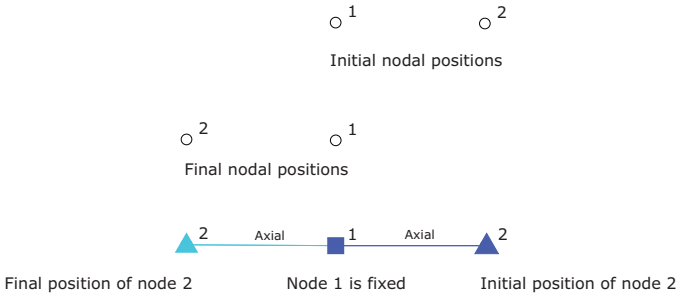


Figure 2: Scenario that could lead to violation of a connector stop specification.

Output

The Abaqus output variables available for connectors are listed in *Abaqus/Standard Output Variable Identifiers* and *Abaqus/Explicit Output Variable Identifiers*. The following output variables are of particular interest when defining stops and locks in connectors:

CSLST	Flags for connector stops and locks.
CRF	Connector reaction forces/moments.

At a given time and for a particular component of relative motion *i*, the output variable CSLST*i* is 1 if the connector is actually stopped or locked in that component (stop or lock criteria are met). In that case, the correspondent CRF output variable will most likely be nonzero and equal to the actual force/moment required to enforce the stop or lock constraint. Since CRF is included in the calculation of CTF, the latter will change as well when the lock or stop is active.

If the stop or lock criteria are not met at a given time for a particular component *i*, the output variable CSLST*i* is 0 and in most cases the correspondent reaction force CRF is zero (the only possible exception is when a connector motion is also applied in that component).

Connector Failure Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Connectors*
- *Connector Behavior*
- **CONNECTOR BEHAVIOR*
- **CONNECTOR FAILURE*
- *Defining failure*

Overview

Connector failure behavior:

- can be defined in any connector with available components of relative motion in Abaqus/Standard;
- can be defined in any connector in Abaqus/Explicit;
- can be used in Abaqus/Standard to fail all or specified components of relative motion if a failure criterion is met;
- can be used in Abaqus/Explicit to fail all or specified components if a failure criterion is met;
- can be triggered if either a connector relative motion or connector force in a specified component is outside a specified range; and
- can be replaced in most cases by the more sophisticated connector damage initiation/evolution behavior (see *Connector Damage Behavior*).

Defining Connector Failure Behavior

A typical connector might have pieces that break if a relative motion component, force, or moment becomes too large. Abaqus provides a way to define which components of relative motion will break and the criteria used to release these components. You can select the component of relative motion on which the failure criterion is based.

In Abaqus/Standard connector failure can be used to specify connector behavior based on available components of relative motion. In Abaqus/Explicit connector failure can be used to specify connector behavior based on constrained as well as available components of relative motion. Limit values for force or moment can be specified for all components of relative motion involved in the connection. In addition, for connectors with available components of relative motion, limit values can be specified for the relative positions corresponding to an available component.

In Abaqus/Standard if the failure criterion specified for the selected component of relative motion is met, either all components of relative motion fail or a single available component fails. By default, all components of relative motion are released upon meeting the failure criterion. The nodal force contributions for all released components from the connector element will be removed during the increment when the failure criterion is met.

In Abaqus/Explicit if the failure criterion specified for the selected component is met, either all components or a single available component fails. By default, all components are released upon meeting the failure criterion. The nodal force contributions for all released components from the connector element will be removed during the increment when the failure criterion is met.

Input File Usage: Use the following options to define connector failure:

*CONNECTOR BEHAVIOR, NAME=*name*
*CONNECTOR FAILURE, COMPONENT=*component number*,
RELEASE=ALL or *component number*

Abaqus/CAE Usage: Interaction module: connector section editor: **Add->Failure**: **Components:** *component or components*, **Release: All** or **Specify** *component*

Viscous Damping in Abaqus/Standard

In Abaqus/Standard the sudden release of the failed connection may lead to convergence problems. To avoid convergence problems, you can add viscous damping to the components. Damping forces in the component are calculated as $\boldsymbol{F_i = \mu v_i}$, where $\boldsymbol{\mu}$ is the user-defined damping coefficient and $\boldsymbol{v_i}$ is the velocity of the failed component. Viscous damping is applied only if a selected available component of relative motion is released.

Input File Usage: Use the following options to add viscous damping to failed components in Abaqus/Standard:

*SECTION CONTROLS, NAME=*name*, VISCOSITY=$\boldsymbol{\mu}$
*CONNECTOR SECTION, CONTROLS=*name*

Abaqus/CAE Usage: Viscous regularization is not supported in Abaqus/CAE.

Example

In the example in *Figure 1* assume that the shock absorber pulls apart if the tensile force in the shock exceeds 800.0 units of force.

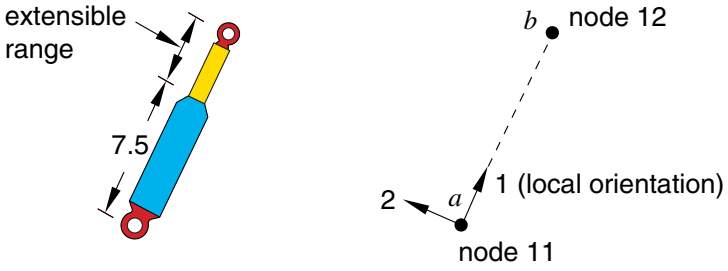


Figure 1: Simplified connector model of a shock absorber.

```
...
*CONNECTOR BEHAVIOR, NAME=sbehavior
*CONNECTOR FAILURE, COMPONENT=1, RELEASE=ALL
, , , 800.0
```

Output

The Abaqus output variables available for connectors are listed in *Abaqus/Standard Output Variable Identifiers* and *Abaqus/Explicit Output Variable Identifiers*. The following output variables are of particular interest when defining failure in connectors:

CFAILST Flags for connector failure status.

ALLVD
Energy dissipated by viscous damping added to failed components.

At any given time and for a particular component of relative motion *i*, the output variable CFAILST*i* is 1 if the connector fails in that particular component of relative motion (failure criteria are met).

If the failure criteria are not met at a given time for a particular component *i*, the output variable CFAILST*i* is 0.

Connector Uniaxial Behavior

Products: Abaqus/Explicit

References:

- *About Connectors*
- *Connection Types*
- *Connector Behavior*
- **CONNECTOR BEHAVIOR*
- **LOADING DATA*
- **UNLOADING DATA*

Overview

Connector uniaxial behavior:

- can be defined in any connector with available components of relative motion by specifying the loading and unloading behavior;
- can be specified for each available component of relative motion independently;
- can define separate response in the tensile and compressive directions;
- can exhibit nonlinear elastic behavior, damaged elastic behavior, or elastic-plastic type behavior with permanent deformation upon complete unloading;
- can have an unloading response specified; and
- can be specified as dependent on constitutive motions in several local directions.

The local directions for each connection type (as described in *Connection Types*) determine the directions in which the forces and moments act and in which the displacements and rotations are measured.

Specifying Uniaxial Behavior for an Available Component of Relative Motion

Uniaxial behavior can be specified for an available component of relative motion by defining the loading and unloading response for that component. For each component, separate loading/unloading response data can be defined for the response in the tensile and compressive directions. The loading and unloading response can be classified according to three available behavior types:

- nonlinear elastic behavior;
- damaged elastic behavior; and
- elastic-plastic type behavior with permanent deformation.

To define the loading response, you specify forces or moments as nonlinear functions of the components of relative motion. These functions can also depend on temperature, field variables, and constitutive displacements/rotations in the other component directions. See *Input Syntax Rules* for further information about defining data as functions of temperature and field variables.

The unloading response can be defined in the following ways:

- You can specify several unloading curves that express the forces or moments as nonlinear functions of the components of relative motion; Abaqus interpolates these curves to create an unloading curve that passes through the point of unloading in an analysis.

- You can specify an energy dissipation factor (and a permanent deformation factor for models with permanent deformation), from which Abaqus calculates an exponential/quadratic unloading function.
- You can specify the forces or moments as nonlinear functions of the components of relative motion, as well as a transition slope; the connector unloads along the specified transition slope until it intersects the specified unloading function, at which point it unloads according to the function. (This unloading definition is referred to as combined unloading.)
- You can specify the forces or moments as nonlinear functions of the components of relative motion; Abaqus shifts the specified unloading function along the strain axis so that it passes through the point of unloading in an analysis.

The behavior type that is specified for the loading response dictates the type of unloading you can define, as summarized in *Table 1*. The different behavior types, as well as the associated loading and unloading curves, are discussed in more detail in the sections that follow.

Table 1: Available unloading definitions for the uniaxial behavior types.

Material behavior type	Unloading definition				
	Interpolated	Quadratic	Exponential	Combined	Shifted
Rate-dependent elastic	√				
Damaged elastic	√	√	√	√	
Permanent deformation	√	√	√		√

Input File Usage: Use the following options to define connector uniaxial behavior:

**CONNECTOR BEHAVIOR*, NAME=*name*
**CONNECTOR UNIAXIAL BEHAVIOR*, COMPONENT=*component number*
**LOADING DATA*, DIRECTION=*deformation direction*,
TYPE=*behavior type*
data lines to define loading data
**UNLOADING DATA*
data lines to define unloading data

Defining the Deformation Direction

The loading/unloading data can be defined separately for tension and compression by specifying the deformation direction. If the deformation direction is defined (tension or compression), the tabular values defining tensile or compressive behavior should be specified with positive values of forces/moments and displacements/rotations in the specified component of relative motion and the loading data must start at the origin. If the behavior is not defined in a loading direction, the force response will be zero in that direction (the connector has no resistance in that direction).

If the deformation direction is not defined, the data apply to both tension and compression. However, the behavior is then considered to be nonlinear elastic and no damage or permanent deformation can be specified. The response data will be considered to be symmetric about the origin if either tensile or compressive data are omitted.

Input File Usage: Use the following option to define tensile behavior:

**LOADING DATA*, DIRECTION=TENSION

Use the following option to define compressive behavior:

*LOADING DATA, DIRECTION=COMPRESSION

Use the following option to define both tensile compressive behavior in a single table:

*LOADING DATA

Behavior That Depends on Relative Positions or Motions in Multiple Component Directions

By default, the loading and unloading functions depend only on the displacement or rotation in the direction of the component of relative motion specified for the connector uniaxial behavior definition (see *Connector Behavior* for details). However, it is also possible to define loading and unloading functions that depend on the constitutive displacements and rotations in multiple component directions.

Input File Usage: Use the following option to define connector uniaxial behavior that depends on the relative displacements and/or rotations in several component directions:

*LOADING DATA, INDEPENDENT COMPONENTS=CONSTITUTIVE MOTION

Defining Rate-Independent Nonlinear Elastic Behavior

When the loading response is rate independent, the unloading response is also rate independent and occurs along the same user-specified loading curve as illustrated in *Figure 1*. An unloading curve does not need to be specified.

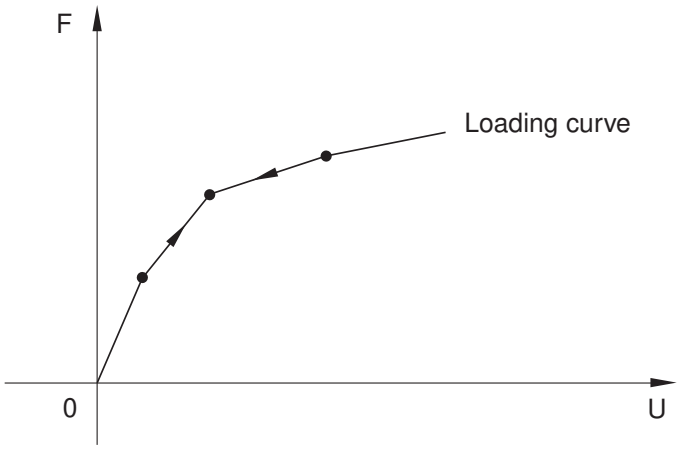


Figure 1: Nonlinear elastic loading.

Input File Usage: *LOADING DATA, TYPE=ELASTIC

Defining Rate-Dependent Behavior

The rate-dependent models require the specification of force-displacement curves at different rates of deformation to describe both loading and unloading behavior. If unloading behavior is not specified, the unloading occurs along the loading curve with the smallest rate of deformation. As the rate of deformation changes, the response is obtained by interpolation of the specified loading/unloading data. Unphysical jumps in the forces due to sudden changes in the rate of deformation are prevented using a technique based on viscoplastic regularization. This

technique also helps model relaxation effects in a very simplistic manner, with the relaxation time given as $\tau = \mu_0 + \mu_1|\lambda - 1|^{\alpha}$ where μ_0, μ_1, and α are material parameters and λ is the stretch. μ_0 is a linear viscosity parameter that controls the relaxation time when $\lambda \approx 1$. Small values of this parameter should be used. μ_1 is a nonlinear viscosity parameter that controls the relaxation time at higher values of λ. The smaller this value, the shorter the relaxation time. α controls the sensitivity of the relaxation speed to the stretch in the component of relative motion. Suggested values of these parameters are $\mu_0 = 0.0001$, $\mu_1 = 0.005$, and $\alpha = 2$. *Figure 2* illustrates the loading/unloading behavior as the connector is loaded at a rate $\dot{u}_2^l$ and then unloaded at a rate $\dot{u}_2^u$.

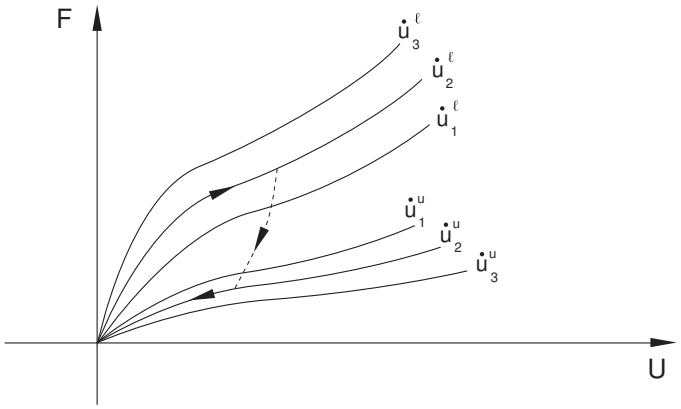


Figure 2: Rate-dependent loading/unloading.

Figure 3 shows the loading/unloading response of a connector element for two different relaxation times τ_1 and τ_2 with $\tau_2 > \tau_1$. The larger the relaxation time, the longer it takes to achieve the specified loading/unloading response for the applied deformation rate.

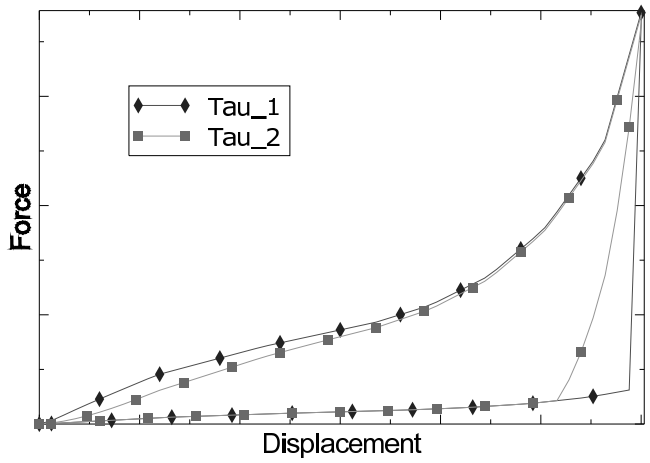


Figure 3: Rate-dependent loading/unloading.

Input File Usage: Use the following options when the unloading is also rate dependent:

**LOADING DATA*, TYPE=ELASTIC, RATE DEPENDENT
**UNLOADING DATA*, DEFINITION=INTERPOLATED CURVE, RATE DEPENDENT

Use the following options when the unloading is rate independent:

*LOADING DATA, TYPE=ELASTIC, RATE DEPENDENT
*UNLOADING DATA, DEFINITION=INTERPOLATED CURVE

Defining Models with Damage

The damage models dissipate energy upon unloading, and there is no permanent deformation upon complete unloading. The unloading behavior controls the amount of energy dissipated by damage mechanisms and can be specified in one of the following ways:

- an analytical unloading curve (exponential/quadratic);
- an unloading curve interpolated from multiple user-specified unloading curves; or
- unloading along a transition unloading curve (constant slope specified by user) to the user-specified unloading curve (combined unloading).

For an overview of the different available behaviors, see *Specifying Uniaxial Behavior for an Available Component of Relative Motion* above. The various unloading types are discussed in the sections that follow.

Defining Onset of Damage

You can specify the onset of damage by defining the displacement below which unloading occurs along the loading curve.

Input File Usage: *LOADING DATA, TYPE=DAMAGE, DAMAGE ONSET=*value*

Specifying Exponential/Quadratic Unloading

The damage model in *Figure 4* is based on an analytical unloading curve that is derived from an energy dissipation factor, H (fraction of energy that is dissipated at any displacement level). As the connector is loaded, the force follows the path given by the loading curve. If the connector is unloaded (for example, at point *B*), the force follows the unloading curve BCO. Reloading after unloading follows the unloading curve OCB until the loading is such that the displacement becomes greater than u_B^{max}, after which the loading path follows the loading curve. The arrows shown in *Figure 4* illustrate the loading/unloading paths of this model.

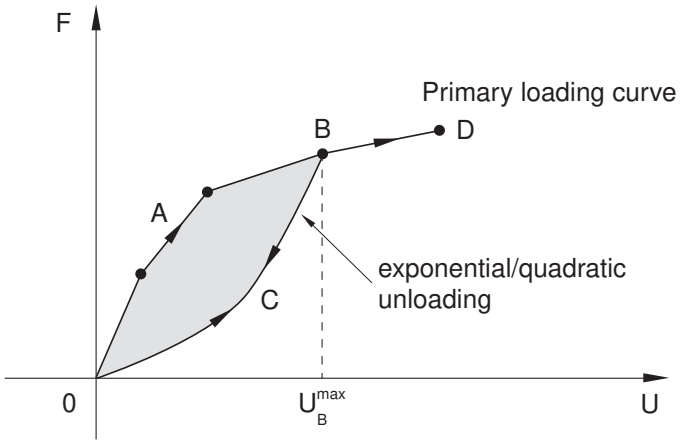


Figure 4: Exponential/quadratic unloading.

The unloading response follows the loading curve when the calculated unloading curve lies above the loading curve to prevent energy generation and follows a zero force response when the unloading curve yields a negative response. In such cases the dissipated energy will be less than the value specified by the energy dissipation factor.

Input File Usage: Use the following option to define quadratic unloading behavior:

**UNLOADING DATA*, DEFINITION=QUADRATIC

Use the following option to define exponential unloading behavior:

**UNLOADING DATA*, DEFINITION=EXPONENTIAL

Specifying Interpolated Curve Unloading

The damage model in *Figure 5* illustrates an interpolated unloading response based on multiple unloading curves that intersect the primary loading curve at increasing values of forces/displacements. You can specify as many unloading curves as are necessary to define the unloading response. Each unloading curve always starts at point *O*, the point of zero force and zero displacements, since the damage models do not allow any permanent deformation. The unloading curves are stored in normalized form so that they intersect the loading curve at a unit force for a unit displacement, and the interpolation occurs between these normalized curves. If unloading occurs from a maximum displacement for which an unloading curve is not specified, the unloading is interpolated from neighboring unloading curves. As the connector is loaded, the force follows the path given by the loading curve. If the connector is unloaded (for example, at point *B*), the force follows the unloading curve BCO. Reloading after unloading follows the unloading path OCB until the loading is such that the displacement becomes greater than u_B^{max}, after which the loading path follows the loading curve.

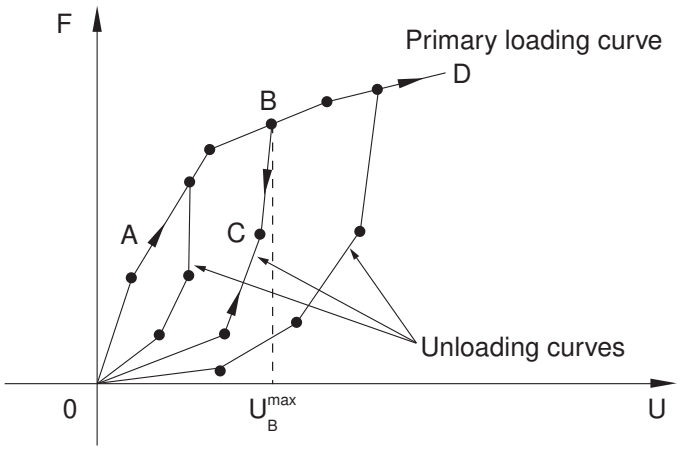


Figure 5: Interpolated curve unloading

If the loading curve depends on the constitutive displacements/rotations in several component directions, the unloading curves also depend on the same component directions. The unloading curves also have the same temperature and field variable dependencies as the loading curve.

Input File Usage: **UNLOADING DATA*, DEFINITION=INTERPOLATED CURVE

Specifying Combined Unloading

As illustrated in *Figure 6*, you can specify an unloading curve OCE in addition to the loading curve $OABD$ as well as a constant transition slope that connects the loading curve to the unloading curve. As the connector is loaded, the force follows the path given by the loading curve. If the connector is unloaded (for example, at point *B*), the force follows the unloading curve BCO. The path BC is defined by the constant transition slope, and CO lies on the specified unloading curve. Reloading after unloading follows the unloading path OCB until the

loading is such that the displacement becomes greater than u_B^{max}, after which the loading path follows the loading curve.

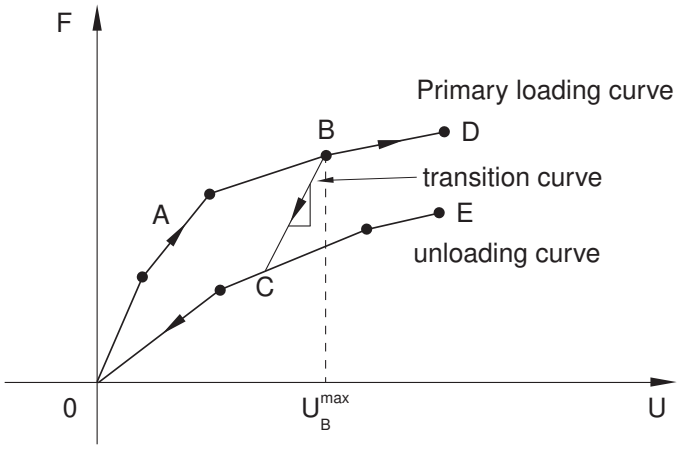


Figure 6: Combined unloading.

If the loading curve depends on the constitutive displacements/rotations in several component directions, the unloading curve also depends on the same component directions. The unloading curve also has the same temperature and field variable dependencies as the loading curve.

Input File Usage: **UNLOADING DATA*, DEFINITION=COMBINED

Defining Models with Permanent Deformation

These models dissipate energy upon unloading and exhibit permanent deformation upon complete unloading. The unloading behavior controls the amount of energy dissipated as well as the amount of permanent deformation. The unloading behavior can be specified in one of the following ways:

- an analytical unloading curve (exponential/quadratic);
- an unloading curve interpolated from multiple user-specified unloading curves; or
- an unloading curve obtained by shifting the user-specified unloading curve to the point of unloading.

For an overview of the different available behaviors, see *Specifying Uniaxial Behavior for an Available Component of Relative Motion* above. The various unloading types are discussed in the sections that follow.

Defining the Onset of Permanent Deformation

By default, the onset of yield will be obtained as soon as the slope of the loading curve decreases by 10% from the maximum slope recorded up to that point while traversing along the loading curve. To override the default method of determining the onset of yield, you can specify either a value for the decrease in slope of the loading curve other than the default value of 10% (slope drop = 0.1) or by defining the displacement below which unloading occurs along the loading curve. If a slope drop is specified, the onset of yield will be obtained as soon as the slope of the loading curve decreases by the specified factor from the maximum slope recorded up to that point.

Input File Usage: Use the following options to specify the onset of yield by defining the displacement below which unloading occurs along the loading curve:

**LOADING DATA*, TYPE=PERMANENT DEFORMATION, YIELD ONSET=*value*

Use the following options to specify the onset of yield by defining a slope drop for the loading curve:

**LOADING DATA*, TYPE=PERMANENT DEFORMATION, SLOPE DROP=*value*

Specifying Exponential/Quadratic Unloading

The model in *Figure 7* illustrates an analytical unloading curve that is derived based on an energy dissipation factor, H (fraction of energy that is dissipated at any displacement level) and a permanent deformation factor, D_p. As the connector is loaded, the force follows the path given by the loading curve. If the connector is unloaded (for example, at point *B*), the force follows the unloading curve BCD. The point *D* corresponds to the permanent deformation, $D_p u_B^{max}$. Reloading after unloading follows the unloading curve DCB until the loading is such that the displacement becomes greater than u_B^{max}, after which the loading path follows the loading curve. The arrows shown in *Figure 7* illustrate the loading/unloading paths of this model.

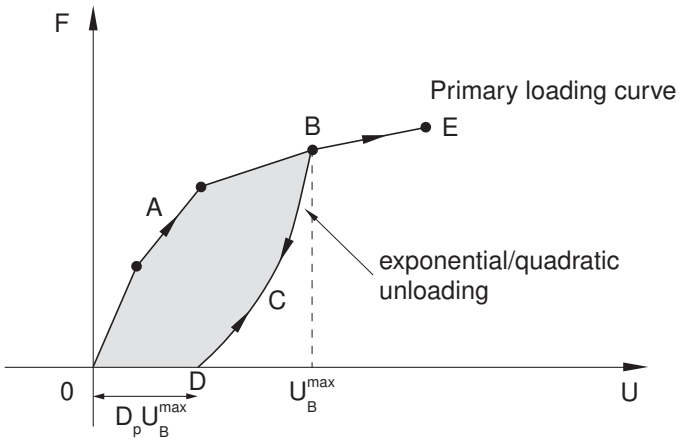


Figure 7: Exponential/quadratic unloading.

The unloading response follows the loading curve when the calculated unloading curve lies above the loading curve to prevent energy generation and follows a zero force response when the unloading curve yields a negative response. In such cases the dissipated energy will be less than the value specified by the energy dissipation factor.

Input File Usage: Use the following option to define quadratic unloading behavior:

**UNLOADING DATA*, DEFINITION=QUADRATIC

Use the following option to define exponential unloading behavior:

**UNLOADING DATA*, DEFINITION=EXPONENTIAL

Specifying Interpolated Curve Unloading

The model in *Figure 8* illustrates an interpolated unloading response based on multiple unloading curves that intersect the primary loading curve at increasing values of forces/displacements. You can specify as many unloading curves as are necessary to define the unloading response. The first point of each unloading curve defines the permanent deformation if the connector is completely unloaded. The unloading curves are stored in normalized form so that they intersect the loading curve at a unit force for a unit displacement, and the interpolation occurs between these normalized curves. If unloading occurs from a maximum displacement for which an unloading curve is not specified, the unloading curve is interpolated from neighboring unloading curves. As the

connector is loaded, the force follows the path given by the loading curve. If the connector is unloaded (for example, at point *B*), the force follows the unloading curve BCD. Reloading after unloading follows the unloading path DCB until the loading is such that the displacement becomes greater than u_B^{max}, after which the loading path follows the loading curve.

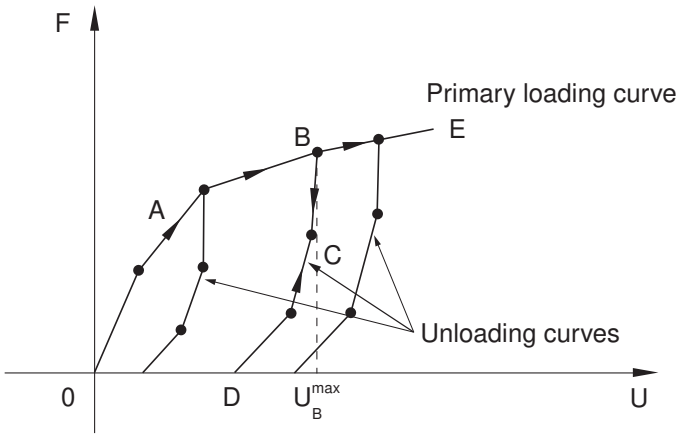


Figure 8: Interpolated curve unloading.

If the loading curve depends on the constitutive displacements/rotations in several component directions, the unloading curves also depends on the same component directions. The unloading curve also has the same temperature and field variable dependencies as the loading curve.

Input File Usage: **UNLOADING DATA*, DEFINITION=INTERPOLATED CURVE

Specifying Shifted Curve Unloading

You can specify an unloading curve passing through the origin in addition to the loading curve. The actual unloading curve is obtained by horizontally shifting the user-specified unloading curve to pass through the point of unloading as shown in *Figure 9*. The permanent deformation upon complete unloading is the horizontal shift applied to the unloading curve.

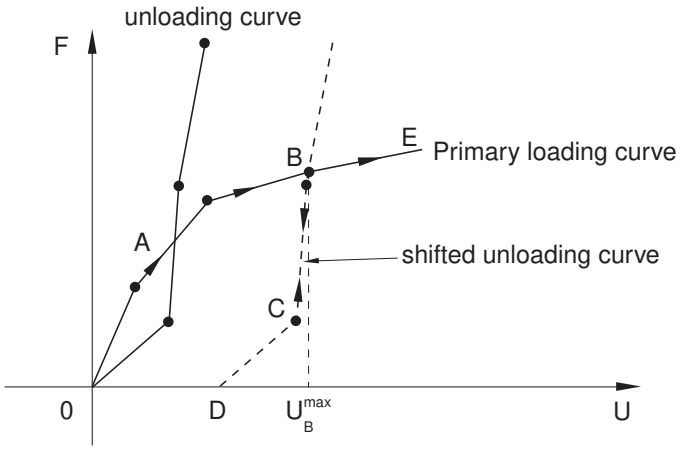


Figure 9: Shifted curve unloading.

If the loading curve depends on the constitutive displacements/rotations in several component directions, the unloading curve also depends on the same component directions. The unloading curve also has the same temperature and field variable dependencies as the loading curve.

Input File Usage: **UNLOADING DATA*, DEFINITION=SHIFTED CURVE

Using Different Uniaxial Models in Tension and Compression

When appropriate, different uniaxial behavior models can be used in tension and compression. For example, a model with permanent deformation and exponential unloading in tension can be combined with a nonlinear elastic model in compression (see *Figure 10*).

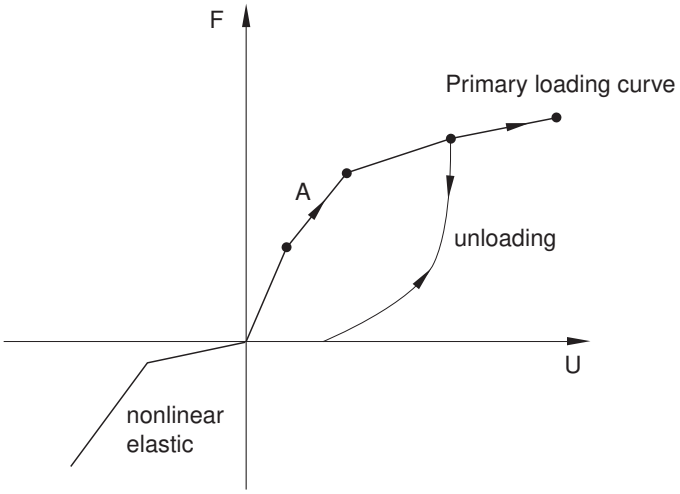


Figure 10: Different uniaxial models in tension and compression.

Output

The Abaqus output variables available for connectors are listed in *Abaqus/Standard Output Variable Identifiers* and *Abaqus/Explicit Output Variable Identifiers*. The following output variables are of particular interest when defining uniaxial behavior in connectors:

CU Connector relative displacements/rotations.

CUF Connector uniaxial forces/moments.

Particle Elements

In this section:

- *Discrete Particle Elements*
- *Continuum Particle Elements*

Discrete Particle Elements

In this section:

- *Discrete Particle Elements*
- *Discrete Particle Element Library*

Discrete Particle Elements

Products: Abaqus/Explicit

References:

- *Discrete Element Method*
- *Discrete Particle Element Library*
- **DISCRETE SECTION*

Overview

Discrete particle elements:

- represent individual particles with a rigid, spherical shape;
- are typically used in analyses with a large number of discrete particle elements interacting with each other and with other bodies;
- can be used only in explicit dynamic analyses; and
- must have only one node.

Typical Applications

Discrete particle elements (PD3D) are useful for simulations involving discontinuous media such as gravel. For a description of the discrete element method (DEM) and its typical applications, see *Discrete Element Method*.

Defining the Elements Section Properties

You must associate a discrete section definition with a set of discrete particle elements. The section definition provides the density associated with the PD3D elements and particle radii.

Input File Usage: **DISCRETE SECTION*, ELSET=*element_set_name*

Particle Geometry, Mass, and Rotational Inertia

PD3D elements are spherical in shape, and each particle has uniform density. The particle radius and density are specified in a discrete section definition. Abaqus uses these quantities to compute the mass and rotational inertia of PD3D elements. Each discrete element is simple, but interactions of large systems of such particles and finite elements enable simulation of complex phenomena, as discussed in *Discrete Element Method*.

Particle sizes and densities will commonly be the same for all particles assigned to a single discrete section definition, but it is possible to specify variations in these quantities among particles associated with a single discrete section definition by assigning distribution names rather than scalar values to the radius and/or density. See *Distribution Definition* for a discussion of distributions. If, for example, two particle sizes are to be considered, it may be most convenient to simply use two discrete section definitions with a single particle size in each.

Input File Usage: Use the following option to define the particle shape, density, and radius:

**DISCRETE SECTION*, SHAPE=SPHERE, DENSITY=*particle density or a particle densities distribution table name*
particle radius or particle radii distribution table name

Alpha Damping

You can define mass and rotary inertia proportional damping for PD3D elements similar to that for point mass and rotary inertia elements (see *Point Masses* and *Rotary Inertia*). This damping acts on translational and rotational velocities of individual particles (with respect to "ground") and is independent of contact damping (which acts on relative velocities for nearby particle pairings). Unlike other types of deformable finite elements in Abaqus/Explicit, there is no global viscous damping acting on PD3D elements. A small amount of mass proportional damping is beneficial in reducing the noise in the solution generated by numerous opening and closing contact conditions.

Input File Usage: Use the following option to specify mass and rotary inertia proportional damping:

**DISCRETE SECTION*, ALPHA=*damping factor,*

Discrete Particle Element Library

Products: Abaqus/Explicit

Overview

This section provides a reference to the particle elements available in Abaqus/Explicit.

References:

- *Discrete Element Method*
- *Discrete Particle Elements*
- **DISCRETE SECTION*

Element Types

Force/Displacement Element

PD3D 1-node discrete particle

Active Degrees of Freedom

1, 2, 3, 4, 5, 6

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage: **DISCRETE SECTION*

Element-Based Loading

Distributed Loads

Gravity loads as described in *Distributed Loads* are the most commonly applied distributed loads for discrete particle elements. You define gravity loading in a specified direction, and the magnitude is input as acceleration.

Element Output

Discrete particle elements have no element output associated with them. Resultant of all contact normal force CNORMF and resultant of all frictional shear force CSHEARF acting on a discrete particle element are the only output variables of interest currently available for this element, see *Abaqus/Explicit Output Variable Identifiers* for details.

Nodes Associated with the Element

1 node.

Continuum Particle Elements

In this section:

- *Continuum Particle Elements*
- *Continuum Particle Element Library*

Continuum Particle Elements

Products: Abaqus/Explicit

References:

- *Smoothed Particle Hydrodynamics*
- *Continuum Particle Element Library*
- **SOLID SECTION*

Overview

Continuum particle elements:

- can be used only in explicit dynamic analyses;
- must have one node only;
- have one integration point;
- can be initialized similarly to continuum elements; and
- are fully filled with material.

Typical Applications

Continuum particle elements (PC3D) are useful for simulations involving material that undergoes extreme deformation such as open-surface fluid flow or obliteration/fragmentation of solid structures. They are defined using only one node; however, the element centered at a given node (particle) receives contributions from all particles within a sphere of influence whose radius is commonly referred to as the smoothing length. The smoothed particle hydrodynamic (SPH) formulation determines at every increment of the analysis the connectivity associated with a given particle. Since nodal connectivity is not fixed, severe element distortion is avoided and, hence, the formulation allows for very high strain gradients.

The 1-node PC3D element is used to define points both on the surface and in the interior of the body to be modeled. You define these nodes similarly to mass elements, and the nodes can be placed in space the same as the nodes of a regular brick mesh. A smoothed particle hydrodynamic mesh is typically a uniformly spaced grid of elements that conforms to the shape of the body being modeled.

For more information, see *Smoothed Particle Hydrodynamics*.

Defining the Elements Section Properties

You must associate a solid section definition with a set of continuum particle elements. The section definition provides the material associated with the PC3D elements.

As part of the solid section definition, you can define a characteristic length. This characteristic length, not to be confused with the smoothing length, is used to compute the particle volume. The volume is assumed to be a cube whose sides are equal to twice the specified characteristic length.

Input File Usage: **SOLID SECTION*, ELSET=*element_set_name*
characteristic length associated with the particle volume

where the ELSET parameter refers to a set of particle elements.

Continuum Particle Element Library

Products: Abaqus/Explicit

Overview

This section provides a reference to the particle elements available in Abaqus/Explicit.

References:

- *Smoothed Particle Hydrodynamics*
- *Continuum Particle Elements*
- **SOLID SECTION*

Element Types

Stress/Displacement Element

PC3D 1-node continuum particle

Active Degrees of Freedom

1, 2, 3

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage: **SOLID SECTION*

Element-Based Loading

Distributed Loads

Gravity loads as described in *Distributed Loads* are the only distributed loads that are available for particle elements. You define gravity loading in a specified direction, and the magnitude is input as acceleration.

Element Output

Output is in global directions unless a local coordinate system is assigned to the element through the section definition (*Orientations*), in which case output is in the local coordinate system (which rotates with the motion in large-displacement analysis). See *State storage* for details.

Stress, Strain, and Other Tensor Components

Stress, strain, and other tensors are available. All tensors have the same components. For example, the stress components are as follows:

S11
: ***XX***, direct stress.

S22
: ***YY***, direct stress.

S33
: ***ZZ***, direct stress.

S12
: ***XY***, shear stress.

S13
: ***XZ***, shear stress.

S23
: ***YZ***, shear stress.

Note: the order shown above is not the same as that used in user subroutine *VUMAT*.

Nodes Associated with the Element

1 node.

Special-Purpose Elements

In this section:

- *Acoustic Interface Elements*
- *Cohesive Elements*
- *Dashpot Elements*
- *Distributing Coupling Elements*
- *Drag Chain Elements*
- *Elastic-Plastic Joints*
- *Eulerian Elements*
- *Flexible Joint Elements*
- *Fluid Pipe Connector Elements*
- *Fluid Pipe Elements*
- *Gasket Elements*
- *Line Spring Elements*
- *Pipe-Soil Elements*
- *Spring Elements*
- *Surface Elements*
- *Tube Support Elements*
- *User-Defined Elements*

Acoustic Interface Elements

In this section:

- *Acoustic Interface Elements*
- *Acoustic Interface Element Library*

Acoustic Interface Elements

Products: Abaqus/Standard Abaqus/CAE

References:

- *Acoustic Interface Element Library*
- *Coupled Acoustic-Structural Analysis*
- **INTERFACE*
- *Creating acoustic interface sections*

Overview

Acoustic interface elements:

- can be used to couple a model of an acoustic fluid to a structural model containing continuum or structural elements;
- couple the accelerations of the surface of the structural model to the pressure in the acoustic medium;
- can be used in dynamic and steady-state dynamic procedures;
- must be defined with the nodes shared by the acoustic elements and the structural (or solid) elements;
- can be used only in small-displacement simulations and are not intended for use in nonlinear or hydrostatic fluid-structure interactions;
- are ignored in eigenfrequency extraction analyses if the subspace iteration eigensolver is used; and
- if necessary, can be degenerated into triangular elements.

For most problems the surface-based, structural-acoustic capabilities described in *Mesh Tie Constraints* and in *Defining Tied Contact in Abaqus/Standard* provide more general and easy to use methods for modeling the interaction between an acoustic fluid and a structure. User-specified acoustic interface elements give you increased control over the coupling specification, at the expense of the convenience of the surface-based procedures.

Typical Applications

The acoustic interface elements are used in simulations where the motion of a solid structure influences the pressure in the acoustic fluid, such as when the vibrations of a car frame produce noise in the passenger compartment; or where the pressure in the fluid affects a neighboring structure, such as when the small-amplitude sloshing of a fluid inside a container affects its response.

User-specified acoustic interface elements are also useful in problems involving only an acoustic medium because they allow you to specify displacement, velocity, or acceleration boundary conditions directly on the nodes of the acoustic interface elements. In this application, however, you must be aware that the tangential displacements are not coupled to the fluid. Therefore, zero-energy modes may arise involving the displacement degrees of freedom if these nodes are not constrained in the tangential direction. When acoustic interface elements are used to couple fluid and solid elements, this problem does not arise because of the stiffness and inertia of the solid.

Choosing an Appropriate Element

The order of the underlying acoustic and structural elements usually dictates which acoustic interface element should be used. The general acoustic interface element, ASI1, can be used in any coupled acoustic-structural simulation; however, normally it is used only with the acoustic link elements (AC1D2 and AC1D3).

Defining the Normal Direction of the Acoustic-Structural Interface

The connectivity of the acoustic interface elements and the right-hand rule define the normal direction of the acoustic-structural interface, as shown in *Acoustic Interface Element Library*. It is very important that this normal point into the acoustic fluid, as shown in *Figure 1* and *Figure 2*. The one exception is the ASI1 acoustic interface element, where you must define the normal direction.

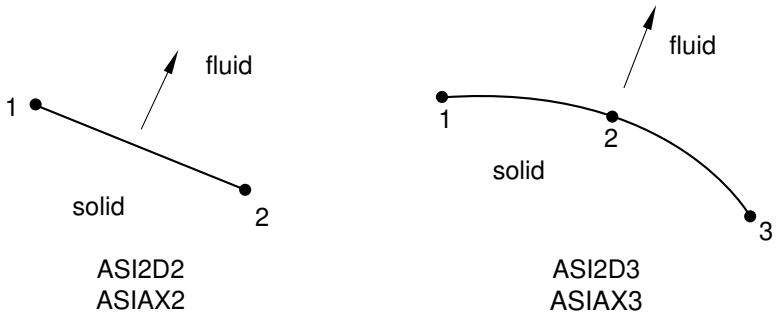


Figure 1: Normal directions for two-dimensional and axisymmetric acoustic-structural interface elements.

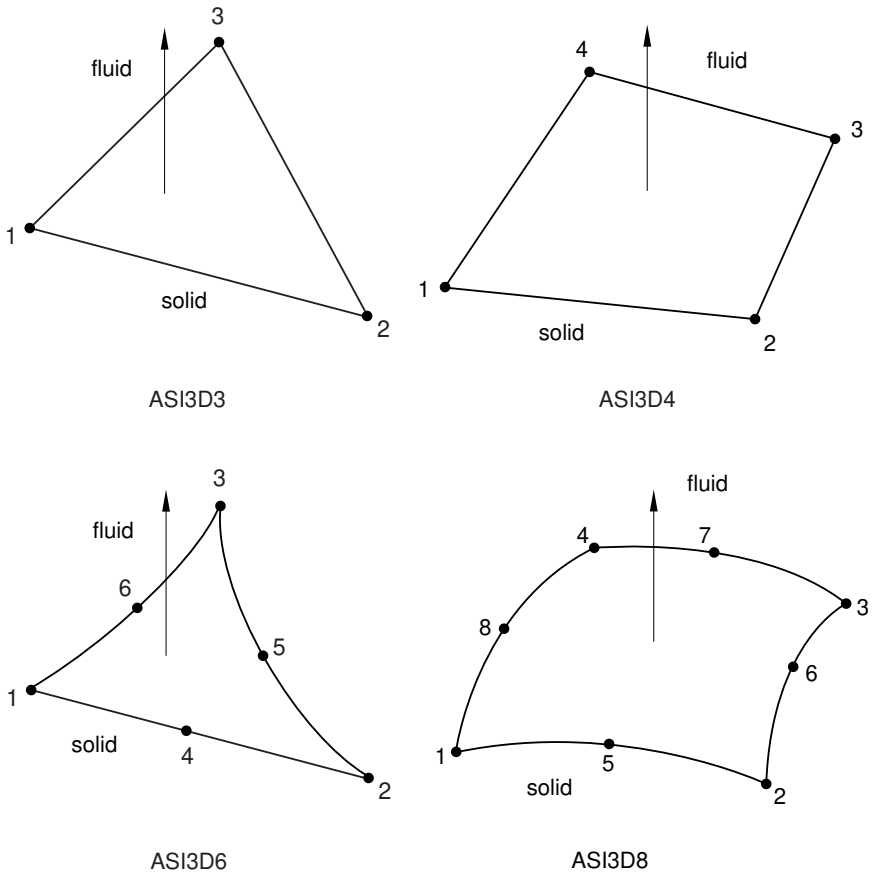


Figure 2: Normal directions for three-dimensional acoustic-structural interface elements.

Defining the Acoustic Interface Element's Section Properties

You must associate the acoustic interface section definition with a set of acoustic interface elements. This section definition must be used with three-dimensional and axisymmetric acoustic interface elements, even though there are no user-defined geometric properties for these elements.

Input File Usage: **INTERFACE*, ELSET=*element_set_name*

Abaqus/CAE Usage: Property module:
Create Section: select **Other** as the section **Category** and **Acoustic interface** as the section **Type**
Assign->Section: select regions

Defining the Geometric Properties Associated with ASI1 Elements

The ASI1 elements consist of a single node. Abaqus/Standard cannot calculate the surface area associated with these elements, so you must supply this information. If accurate surface areas are not given, Abaqus/Standard may calculate incorrect accelerations or acoustic fluid pressure at the acoustic-structural interface.

In addition, Abaqus/Standard cannot calculate the direction of the interface normal associated with these elements. You must provide the direction cosines, in the global Cartesian coordinate system, of the interface normal for these elements.

Input File Usage: **INTERFACE*
surface area, *X-direction cosine*, *Y-direction cosine*, *Z-direction cosine*

Abaqus/CAE Usage: General-use acoustic interface sections are not supported in Abaqus/CAE.

Defining the Thickness for Planar Acoustic Interface Elements

You can specify the thickness of planar acoustic interface elements. The default value is unit thickness.

Input File Usage: **INTERFACE*
thickness

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Acoustic interface** as the section **Type**: **Plane stress/strain thickness**: *thickness*

Using Acoustic Interface Elements When Elements with Different Interpolation Orders Form the Acoustic-Structural Interface

It is normally assumed that the same order of interpolation will be used for both the acoustic fluid mesh and the structural mesh (at least at the interface surfaces). If this is not the case, suitable MPCs must be applied to the nodes along the acoustic-structural interface to maintain the compatibility in the pressure (MPC type P LINEAR) or displacement fields (MPC type LINEAR).

Acoustic Interface Element Library

Products: Abaqus/Standard Abaqus/CAE

Overview

This section provides a reference to the acoustic interface elements available in Abaqus/Standard.

References:

- *Acoustic Interface Elements*
- **INTERFACE*

Element Types

Element for General Use

ASI1 1-node

Active Degrees of Freedom

1, 2, 3, 8

Additional Solution Variables

None.

Elements for Use in Planar Models

ASI2D2 2-node linear

ASI2D3 3-node quadratic

Active Degrees of Freedom

1, 2, 8

Additional Solution Variables

None.

Elements for Use in 3D Models

ASI3D3 3-node linear

ASI3D4	4-node linear
ASI3D6	6-node quadratic
ASI3D8	8-node quadratic

Active Degrees of Freedom

1, 2, 3, 8

Additional Solution Variables

None.

Elements for Use in Axisymmetric Models

ASIAX2	2-node linear
ASIAX3	3-node quadratic

Active Degrees of Freedom

1, 2, 8

Additional Solution Variables

None.

Nodal Coordinates Required

General use element: None.

Planar: *X*, *Y*

3D: *X*, *Y*, *Z*

Axisymmetric: *r*, *z*

Element Property Definition

For general-use elements, you must define the element's surface area and the direction cosines of the normal to the acoustic fluid-structural interface, pointing into the fluid.

For elements for use in planar models, you must specify the thickness (out-of-plane) of the element. The default is unit thickness if no thickness is specified.

For elements for use in three-dimensional and axisymmetric models, no additional data are required.

Input File Usage: **INTERFACE*

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Acoustic interface** as the section **Type**

General-use acoustic interface sections are not supported in Abaqus/CAE.

Element-Based Loading

Distributed impedances cannot be applied.

Element Output

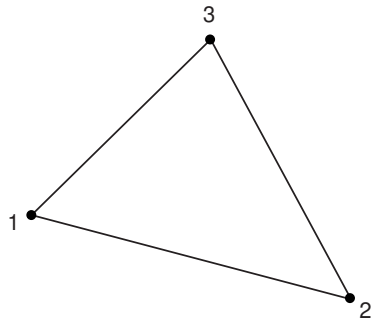
None.

Node Ordering on Elements

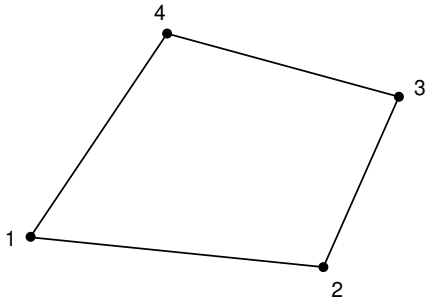
Planar



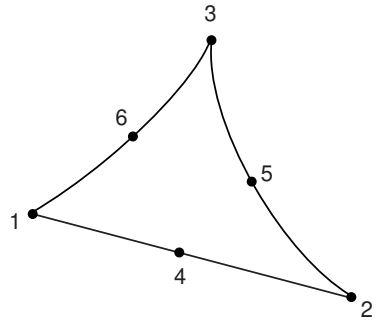
3D



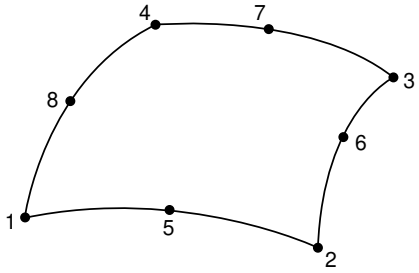
ASI3D3



ASI3D4

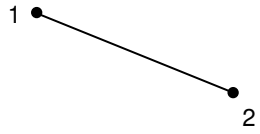


ASI3D6

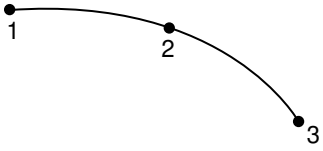


ASI3D8

Axisymmetric



ASIAX2



ASIAX3

Cohesive Elements

In this section:

- *About Cohesive Elements*
- *Choosing a Cohesive Element*
- *Modeling with Cohesive Elements*
- *Defining the Cohesive Element's Initial Geometry*
- *Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach*
- *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*
- *Defining the Constitutive Response of Fluid within the Cohesive Element Gap*
- *Defining the Constitutive Response of Fluid Transitioning from Darcy Flow to Poiseuille Flow*
- *Defining Thermal Convection for a Cohesive Element*
- *Defining a Thermal Interaction for a Cohesive Element*
- *Defining Slurry Transport and Placement within the Cohesive Element Gap*
- *Two-Dimensional Cohesive Element Library*
- *Three-Dimensional Cohesive Element Library*
- *Axisymmetric Cohesive Element Library*

About Cohesive Elements

Overview

Modeling with cohesive elements consists of:

- choosing the appropriate cohesive element type (*Choosing a Cohesive Element*);
- including the cohesive elements in a finite element model, connecting them to other components, and understanding typical modeling issues that arise during modeling using cohesive elements (*Modeling with Cohesive Elements*);
- defining the initial geometry of the cohesive elements (*Defining the Cohesive Element's Initial Geometry*); and
- defining the mechanical, and optionally the fluid and/or thermal, constitutive behavior of the cohesive elements.

The mechanical constitutive behavior of the cohesive elements can be defined:

- with a continuum-based constitutive model (*Modeling of an Adhesive Layer of Finite Thickness*),
- with a uniaxial stress-based constitutive model useful in modeling gaskets and/or single adhesive patches (*Modeling of Gaskets and/or Small Adhesive Patches*), or
- by using a constitutive model specified directly in terms of traction versus separation (*Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*).

When pore pressure cohesive elements or coupled temperature-pore pressure elements are used in soils procedures in Abaqus/Standard, the fluid constitutive behavior of the cohesive elements can be defined (*Defining the Constitutive Response of Fluid within the Cohesive Element Gap* and *Defining the Constitutive Response of Fluid Transitioning from Darcy Flow to Poiseuille Flow*):

- by defining the tangential fluid flow relationship, and
- by defining a fluid leak-off coefficient that accounts for caking or fouling effects in rock fracture.

When coupled temperature-pore pressure elements are used, the gap fluid heat convection behavior can also be defined (*Defining Thermal Convection for a Cohesive Element*):

- by defining gap fluid density, specific heat, conductivity, and
- by defining gap convection between the gap fluid and both top and bottom surfaces.

When coupled temperature-displacement elements are used in a fully coupled thermal procedure in Abaqus/Standard, the thermal interaction constitutive behavior of the cohesive elements can be defined (*Defining a Thermal Interaction for a Cohesive Element*):

- by defining gap conductance, and
- by defining gap radiation between surfaces when the gap is small.

When slurry concentration cohesive elements are used in a soils procedure in Abaqus/Standard, the slurry transportation and placement within the cohesive elements can be defined (*Defining a Thermal Interaction for a Cohesive Element*) by defining:

- slurry density, specific heat, conductivity, viscosity, diffusivity;
- particle settling velocity; and
- static or dynamic filtration models for fluid leakoff.

Abaqus offers a library of cohesive elements to model the behavior of adhesive joints, interfaces in composites, and other situations where the integrity and strength of interfaces may be of interest.

References:

Typical Applications

Cohesive elements are useful in modeling adhesives, bonded interfaces, gaskets, and rock fracture. The constitutive response of these elements depends on the specific application and is based on certain assumptions about the deformation and stress states that are appropriate for each application area. The nature of the mechanical constitutive response might broadly be classified to be based on:

- a continuum description of the material;
- a traction-separation description of the interface; or
- a uniaxial stress state appropriate for modeling gaskets and/or laterally unconstrained adhesive patches.

Each of these constitutive response types is discussed briefly below.

Continuum-Based Modeling

The modeling of adhesive joints involves situations where two bodies are connected together by a glue-like material (see *Figure 1*). A continuum-based modeling of the adhesive is appropriate when the glue has a finite thickness. The macroscopic properties, such as stiffness and strength, of the adhesive material can be measured experimentally and used directly for modeling purposes (see *Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach* for details). The adhesive material is generally more compliant than the surrounding material. The cohesive elements model the initial loading, the initiation of damage, and the propagation of damage leading to eventual failure in the material.

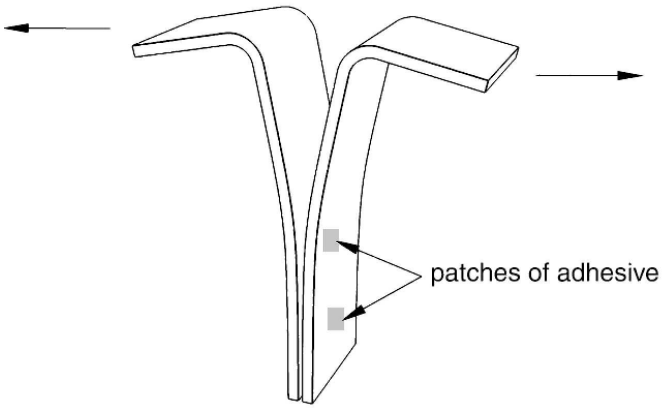


Figure 1: Typical peel test using cohesive elements to model finite-thickness adhesives.

In three-dimensional problems the continuum-based constitutive model assumes one direct (through-thickness) strain, two transverse shear strains, and all (six) stress components to be active at a material point. In two-dimensional problems it assumes one direct (through-thickness) strain, one transverse shear strain, and all (four) stress components to be active at a material point.

Traction-Separation-Based Modeling

The modeling of bonded interfaces in composite materials often involves situations where the intermediate glue material is very thin and for all practical purposes might be considered to be of zero thickness (see *Figure 2*).

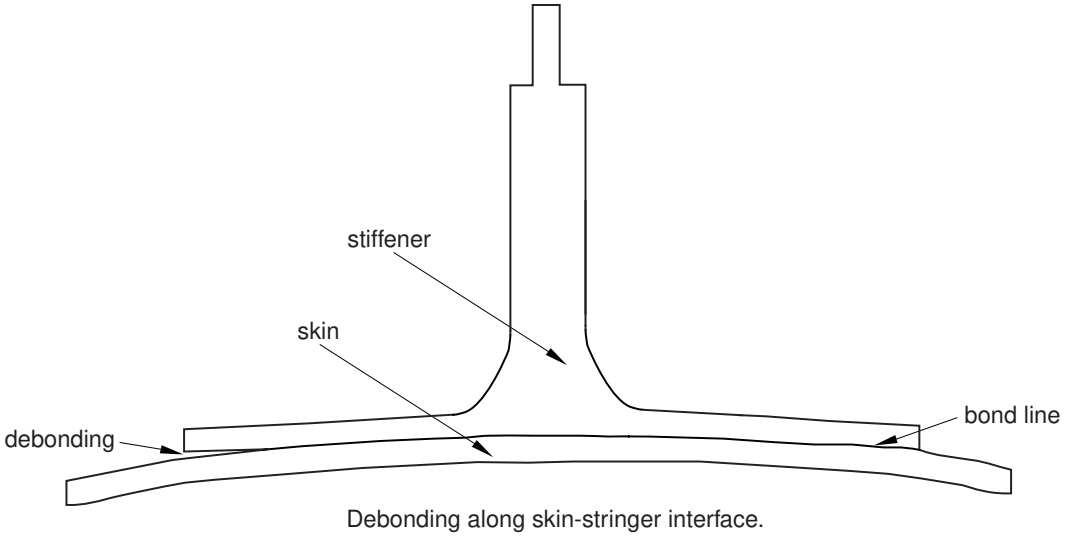


Figure 2: Debonding along a skin-stringer interface: typical situation for traction-separation-based modeling.

In this case the macroscopic material properties are not relevant directly, and the analyst must resort to concepts derived from fracture mechanics—such as the amount of energy required to create new surfaces (see *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description* for details). The cohesive elements model the initial loading, the initiation of damage, and the propagation of damage leading to eventual failure at the bonded interface. The behavior of the interface prior to initiation of damage is often described as linear elastic in terms of a penalty stiffness that degrades under tensile and/or shear loading but is unaffected by pure compression.

You may use the cohesive elements in areas of the model where you expect cracks to develop. However, the model need not have any crack to begin with. In fact, the precise locations (among all areas modeled with cohesive elements) where cracks initiate, as well as the evolution characteristics of such cracks, are determined as part of the solution. The cracks are restricted to propagate along the layer of cohesive elements and will not deflect into the surrounding material.

In three-dimensional problems the traction-separation-based model assumes three components of separation—one normal to the interface and two parallel to it; and the corresponding stress components are assumed to be active at a material point. In two-dimensional problems the traction-separation-based model assumes two components of separation—one normal to the interface and the other parallel to it; and the corresponding stress components are assumed to be active at a material point.

Modeling of Gaskets and/or Laterally Unconstrained Adhesive Patches

Cohesive elements also provide some limited capabilities for modeling gaskets (see *Figure 3*).

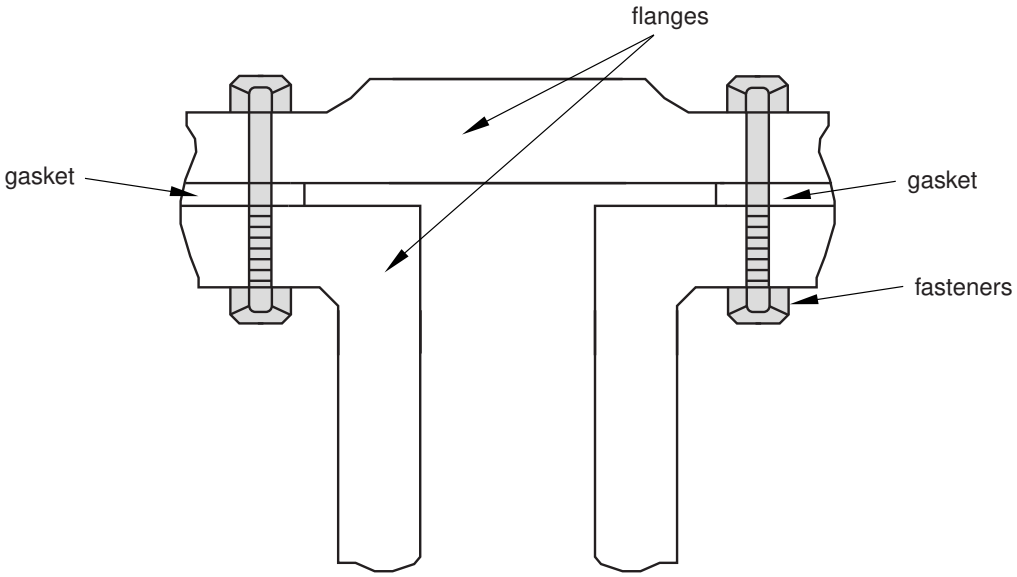


Figure 3: Typical application involving gaskets.

The constitutive response of gaskets modeled with cohesive elements can be defined using only macroscopic properties such as stiffness and strength (see *Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach* for details). No specialized gasket behavior (typically defined in terms of pressure versus closure) is available. Compared to the class of gasket elements available in Abaqus/Standard (*About Gasket Elements*), the cohesive elements

- are fully nonlinear (can be used with finite strains and rotations);
- can have mass in a dynamic analysis; and
- are available in both Abaqus/Standard and Abaqus/Explicit.

It is assumed that the gaskets are subjected to a uniaxial stress state. A uniaxial stress state is also appropriate for modeling small adhesive patches that are unconstrained in the lateral direction.

Any material model in Abaqus that is available for use with a one-dimensional element (beams, trusses, or rebars)—including, for example, the hyperelastic and the elastomeric foam material models (useful in this context for modeling gaskets, sealants, or shock absorbers made out of poron)—can be used with this approach.

Spatial Representation of a Cohesive Element

Figure 4 demonstrates the key geometric features that are used to define cohesive elements.

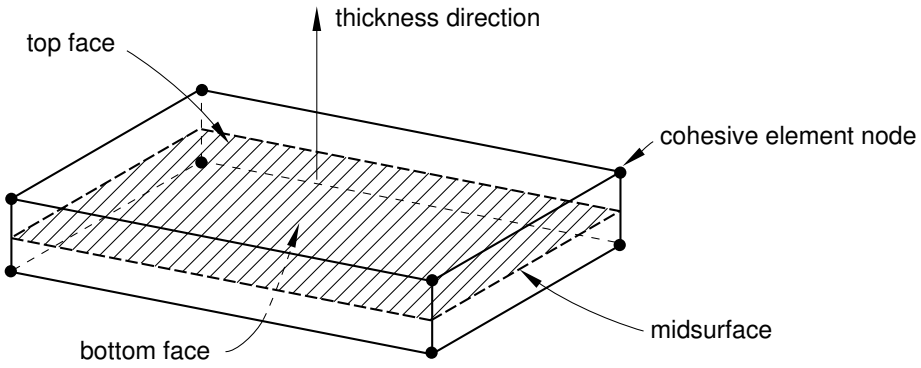


Figure 4: Spatial representation of a three-dimensional cohesive element.

The connectivity of cohesive elements is like that of continuum elements, but it is useful to think of cohesive elements as being composed of two faces separated by a thickness. The relative motion of the bottom and top faces measured along the thickness direction (local 3-direction for three-dimensional elements; local 2-direction for two-dimensional elements—see *Defining the Cohesive Element's Initial Geometry* for further details on local directions) represents opening or closing of the interface. The relative change in position of the bottom and top faces measured in the plane orthogonal to the thickness direction quantifies the transverse shear behavior of the cohesive element. Stretching and shearing of the midsurface of the element (the surface halfway between the bottom and top faces) are associated with membrane strains in the cohesive element; however, it is assumed that the cohesive elements do not generate any stresses in a purely membrane response. *Figure 5* shows the different deformation modes of a cohesive element.

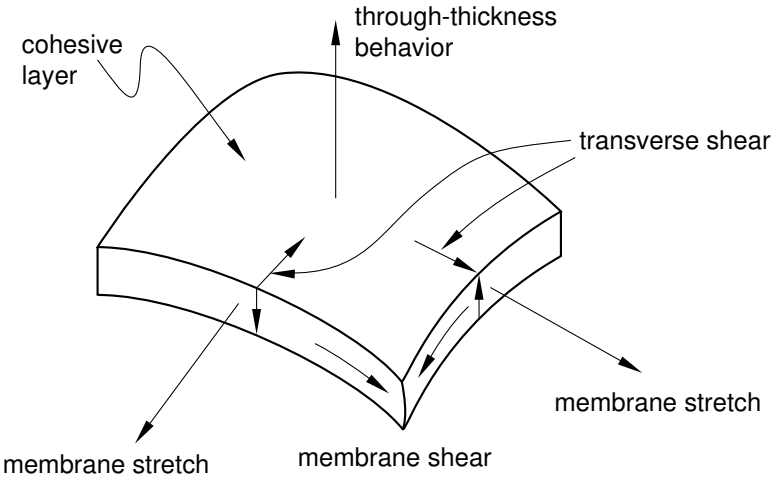


Figure 5: Deformation modes of a cohesive element.

General Issues Related to Modeling with Cohesive Elements

While using cohesive elements, you should be mindful of important issues that are specific to these elements. Such issues include special considerations associated with using cohesive elements in conjunction with contact interactions, potential degradation of the stable time increment size in Abaqus/Explicit, and potential convergence problems in Abaqus/Standard. These issues are discussed in detail in *Modeling with Cohesive Elements*. Cohesive elements are typically used to bond components together. *Modeling with Cohesive Elements* also discusses methods for connecting a cohesive layer to adjacent components.

Procedures with Which Cohesive Elements Are Allowed

Cohesive elements without pore pressure degrees of freedom can be used in all stress/displacement analysis types. Although they do not have any degrees of freedom other than displacement, they can be used in coupled procedures to bond together components made out of coupled temperature-displacement elements, and in Abaqus/Standard coupled pore pressure-displacement elements and/or piezoelectric elements, to simulate mechanical failure of interfaces. The response of the cohesive element in such coupled procedures is mechanical only (for example, no heat transfer occurs across the interface in a coupled temperature-displacement problem).

Cohesive elements with pore pressure degrees of freedom can be used in coupled pore fluid diffusion/stress analyses (*Coupled Pore Fluid Diffusion and Stress Analysis*). The mechanical response of the coupled pore pressure–displacement element is the same as the equivalent displacement-only element, except that the gap fluid pressure is considered as a traction on open faces.

Cohesive elements with temperature degrees of freedom can be used in fully coupled thermal-stress analyses (*Fully Coupled Thermal-Stress Analysis*). The mechanical response of the coupled pore pressure–displacement

element is the same as the equivalent displacement-only element. In addition, the gap thermal interaction is considered between the top and bottom surfaces of a cohesive element.

Cohesive elements with slurry concentration degrees of freedom can be used in coupled pore fluid diffusion/stress analyses (*Coupled Pore Fluid Diffusion and Stress Analysis*). The mechanical response of the coupled pore pressure–displacement element is the same as the equivalent displacement-only element, except that the gap fluid pressure is considered as a traction on open faces. The particle transportation is localized entirely within a fracture.

Choosing a Cohesive Element

Products: Abaqus/Standard Abaqus/Explicit

References:

- *About Cohesive Elements*
- *Two-Dimensional Cohesive Element Library*
- *Three-Dimensional Cohesive Element Library*
- *Axisymmetric Cohesive Element Library*

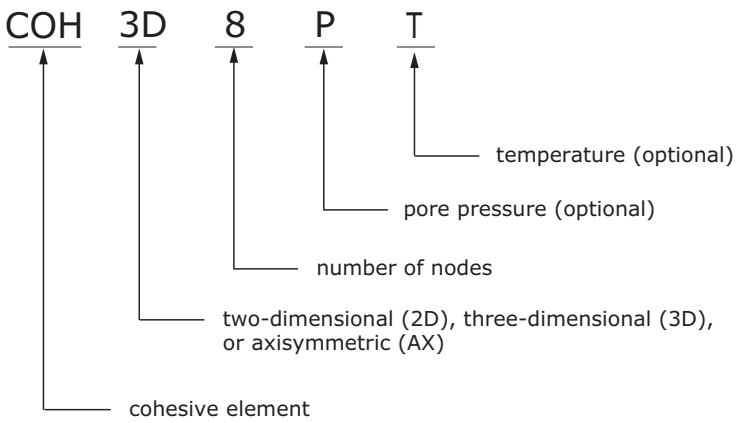
Overview

The Abaqus cohesive element library includes:

- elements for two-dimensional analyses;
- elements for three-dimensional analyses; and
- elements for axisymmetric analyses.

Naming Convention

The cohesive elements used in Abaqus are named as follows:



For example, COH2D4 is a 4-node, two-dimensional cohesive element. However, for pore pressure, coupled thermal pore-pressure, and coupled slurry-thermal-pore pressure cohesive elements , there is an additional layer of nodes on the element's middle face. For example, COH2D4P is a 6-node element, with four corner nodes and two additional nodes along the middle face. For pore pressure cohesive elements that model the transition from Darcy flow to Poiseuille flow, the first three letters for the elements' names are changed to COD. For example, COD2D4P is the 6-node, two-dimensional pore pressure cohesive element with transitional modeling from Darcy flow to Poiseuille flow.

Modeling with Cohesive Elements

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Cohesive Elements*
- *Choosing a Cohesive Element*
- **COHESIVE SECTION*
- *Adhesive joints and bonded interfaces*

Overview

Cohesive elements:

- are used to model adhesives between two components, each of which may be deformable or rigid;
- are used to model interfacial debonding using a cohesive zone framework;
- are used to model gaskets and/or small adhesive patches;
- can be connected to the adjacent components by sharing nodes, by using mesh tie constraints, or by using MPCs type TIE or PIN; and
- may interact with other components via contact for gasket applications.

This section discusses the techniques that are available to discretize cohesive zones and assemble them in a model representing several components that are bonded to one another. It also discusses several common modeling issues related to cohesive elements.

Discretizing Cohesive Zones Using Cohesive Elements

The cohesive zone must be discretized with a single layer of cohesive elements through the thickness. If the cohesive zone represents an adhesive material with a finite thickness, the continuum macroscopic properties of this material can be used directly for modeling the constitutive response of the cohesive zone. Alternatively, if the cohesive zone represents an infinitesimally thin layer of adhesive at a bonded interface, it may be more relevant to define the response of the interface directly in terms of the traction at the interface versus the relative motion across the interface. Finally, if the cohesive zone represents a small adhesive patch or a gasket with no lateral constraint, a uniaxial stress state provides a good approximation to the state of these elements. Abaqus provides modeling capabilities for all the above cases. The details are discussed in later sections.

Connecting Cohesive Elements to Other Components

At least one of either the top or the bottom face of the cohesive element must be constrained to another component. In most applications it is appropriate to have both faces of the cohesive elements tied to neighboring components. If only one face of the cohesive element is constrained and the other face is free, the cohesive element exhibits one or (for three-dimensional elements) more singular modes of deformation due to the lack of membrane stiffness. The singular modes can propagate from one cohesive element to the adjacent one but can be suppressed by constraining the nodes on the side face at the end of a series of cohesive elements.

In some cases it may be convenient and appropriate to have cohesive elements share nodes with the elements on the surfaces of the adjacent components. More generally, when the mesh in the cohesive zone is not matched to the mesh of the adjacent components, cohesive elements can be tied to other components. When cohesive elements are used to model gaskets, it may be more appropriate to tie or share nodes on one side and define contact on the other side as discussed below. This will prevent the gaskets from being subjected to tensile stresses.

Having Cohesive Elements Share Nodes with Other Elements

When the cohesive elements and their neighboring parts have matched meshes, it is straightforward to connect cohesive elements to other components in a model simply by sharing nodes (see *Figure 1*).

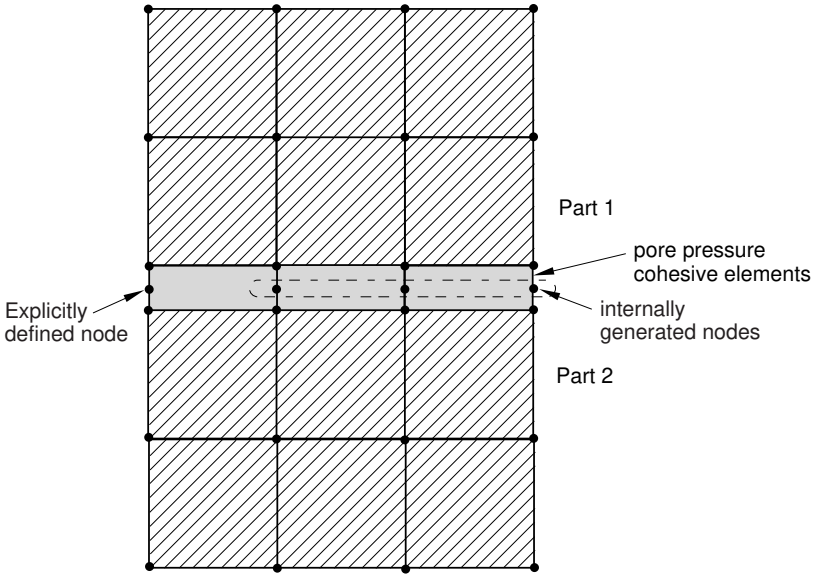


Figure 1: Cohesive elements sharing nodes with other Abaqus elements.

When these elements are used as adhesives or to model debonding, this method can be used to obtain initial results from a model—more accurate local results (in the decohesion zone) would typically be obtained with the cohesive zone more refined than the elements of the surrounding components. When these elements are used to model gaskets, this approach is suitable in situations when no frictional slip occurs between the gaskets and the surrounding components. The method of sharing nodes in gasket applications will lead to tensile stresses in the gasket should the parts connected to the gasket be pulled apart. Defining contact on one side of the cohesive elements will avoid such tensile stresses.

Connecting Cohesive Elements to Other Components by Using Surface-Based Tie Constraints

If the two neighboring parts do not have matched meshes, such as when the discretization level in the cohesive layer is different (typically finer) from the discretization level in the surrounding structures, the top and/or bottom surfaces of the cohesive layer can be tied to the surrounding structures using a tie constraint (*Mesh Tie Constraints*). *Figure 2* shows an example in which a finer discretization is used for the cohesive layer than for the neighboring parts.

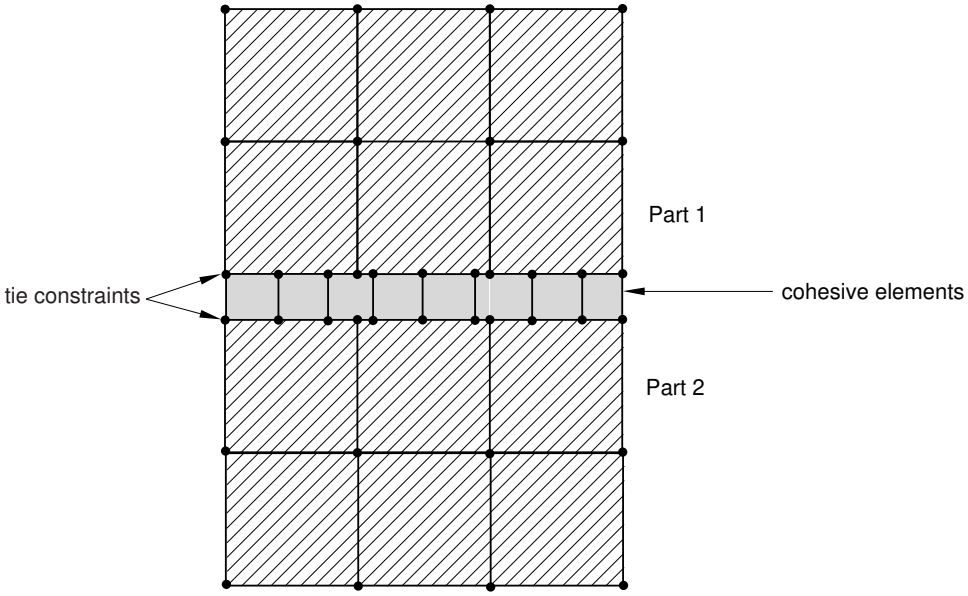


Figure 2: Independent meshes with tie constraints.

Contact Interactions between Cohesive Elements and Other Components

For some applications involving gaskets it is appropriate to define contact on one side of the cohesive element (see *Figure 3*).

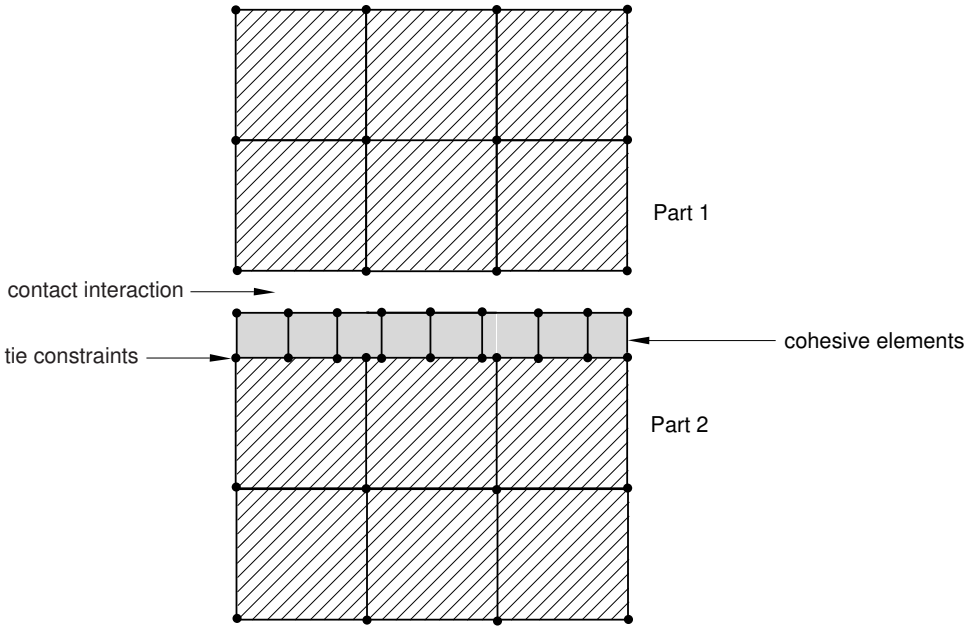


Figure 3: Contact interaction on one side of a cohesive zone.

Contact can be defined with either the general contact algorithm in Abaqus/Explicit (*About General Contact in Abaqus/Explicit*) or the contact pair algorithm in Abaqus/Standard (*About Contact Pairs in Abaqus/Standard*) or Abaqus/Explicit (*About Contact Pairs in Abaqus/Explicit*). If pure main-secondary contact is used, typically the surface of the cohesive elements should be the secondary surface and the surface of the neighboring part should be the main surface. This choice of main and secondary is based on the cohesive zone typically being composed of softer materials and having a finer discretization. The second consideration also suggests that

mismatched meshes will often be used in analyses involving cohesive elements. If mismatched meshes are used, the pressure distribution on the cohesive elements may not be predicted accurately; submodeling (*About Submodeling*) may be required to obtain accurate local results.

Using Cohesive Elements in Large-Displacement Analyses

Cohesive elements can be used in large-displacement analyses. The assembly containing the cohesive elements can undergo finite displacement as well as finite rotation.

Selecting the Broad Class of the Constitutive Response of Cohesive Elements

As discussed earlier, cohesive elements can be used to model finite-thickness adhesives, negligibly thin adhesive layers for debonding applications, as well as gaskets and/or small adhesive patches. You must choose one of these broad classes of applications when you define the section properties of cohesive elements. The detailed implications of each choice are discussed in *Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach* and *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*.

Input File Usage: Use the following option to model a finite-thickness adhesive layer using a continuum-based constitutive response:

**COHESIVE SECTION*, RESPONSE=CONTINUUM

Use the following option to model a negligibly (geometrically) thin layer of adhesive using a traction-separation-based response:

**COHESIVE SECTION*, RESPONSE=TRACTION SEPARATION

Use the following option to use cohesive elements as gaskets and/or small adhesive patches:

**COHESIVE SECTION*, RESPONSE=GASKET

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Cohesive** as the section **Type**: **Response**: **Continuum**, **Traction Separation**, or **Gasket**

Assigning a Material Behavior to a Cohesive Element

You assign the name of a material definition to a particular element set. The constitutive behavior for this element set is defined entirely by the constitutive thickness of the cohesive layer (discussed in *Specifying the Constitutive Thickness*) and the material properties referring to the same name.

The constitutive behavior of the cohesive elements can be defined either in terms of a material model provided in Abaqus or a user-defined material model (see *User-Defined Mechanical Material Behavior*). When cohesive elements are used in applications involving a finite-thickness adhesive, any available material model in Abaqus, including material models for progressive damage, can be used. For applications involving gasket and/or small finite-thickness adhesive patches, any material model that can be used with one-dimensional elements (such as beams, trusses, and rebars), including material models for progressive damage, can be used. For further details, see *Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach*. For applications in which the behavior of cohesive elements is defined directly in terms of traction versus separation, the response can be defined only in terms of a linear elastic relation (between the traction and the separation) along with progressive damage (see *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*).

To define the constitutive behavior of cohesive elements, you assign the name of a material model to a particular element set through the section definition. The actual material model for a user-defined material model is defined in user subroutine *UMAT* in Abaqus/Standard or *VUMAT* in Abaqus/Explicit.

Input File Usage: **COHESIVE SECTION*, ELSET=*name*, MATERIAL=*name*

Abaqus/CAE Usage: Property module: cohesive section editor: **Material**: *name*

Using Cohesive Elements in Coupled Pore Fluid Diffusion/Stress Analyses

Cohesive elements with, or without, pore pressure degrees of freedom can be used in coupled pore fluid diffusion/stress analyses. Cohesive elements without pore pressure degrees of freedom will only contribute mechanically, and surfaces exposed when cohesive elements open will be impermeable to fluid flow.

Cohesive elements with pore pressure degrees of freedom provide a more general response, including the ability to model tangential flow and leakage flow from the gap into the adjacent material. These elements have additional pore pressure nodes in the gap interior, and you can choose to define these nodes explicitly or have them generated automatically by Abaqus/Standard.

In a typical use you will have these gap interior nodes generated for you for the majority of cohesive elements in the model. You invoke automatic node generation as discussed in *By Defining the Bottom-Face Element Connectivity and an Integer Offset*.

Defining Contact Between Surrounding Components

Cohesive elements are used to bond two different components. Often the cohesive elements completely degrade in tension and/or shear as a result of the deformation. Subsequently, the components that are initially bonded together by cohesive elements may come into contact with each other. Several approaches are available for modeling this kind of contact.

Allowing the Cohesive Element to Handle the Contact

In certain situations this kind of contact can be handled by the cohesive element itself. By default, cohesive elements retain their resistance to compression even if their resistance to other deformation modes is completely degraded. As a result, the cohesive elements resist interpenetration of the surrounding components even after the cohesive element has completely degraded in tension and/or shear. This approach works best when the top and the bottom faces of the cohesive element do not displace tangentially by a significant amount relative to each other during the deformation. In other words, to model the situation described above, the deformation of the cohesive elements should be limited to “small sliding.”

Defining Contact and Deleting Cohesive Elements When They Are Completely Degraded

Another possible approach is to define contact between the surfaces of the surrounding components that could potentially come into contact and to delete the cohesive elements once they are completely damaged. Thus, contact is modeled throughout the analysis. This approach is not recommended if the geometric thickness of the cohesive elements in the model is very small or zero (the geometric thickness of the cohesive elements may be different from the constitutive thickness you specify while defining the section properties of the cohesive elements—see *Specifying the Constitutive Thickness*) because contact will effectively cause nonphysical resistance to compression of the cohesive layer while the cohesive elements are still active. If frictional contact is modeled, there may also be nonphysical shearing forces.

This is the behavior that will occur by default with the general contact algorithm in Abaqus/Explicit. *Figure 4*, *Figure 5*, and *Figure 6* show the default surface for general contact. This surface:

- is insensitive to whether the cohesive elements and neighboring elements share nodes, are tied together, or are not connected; and
- does not include faces of cohesive elements.

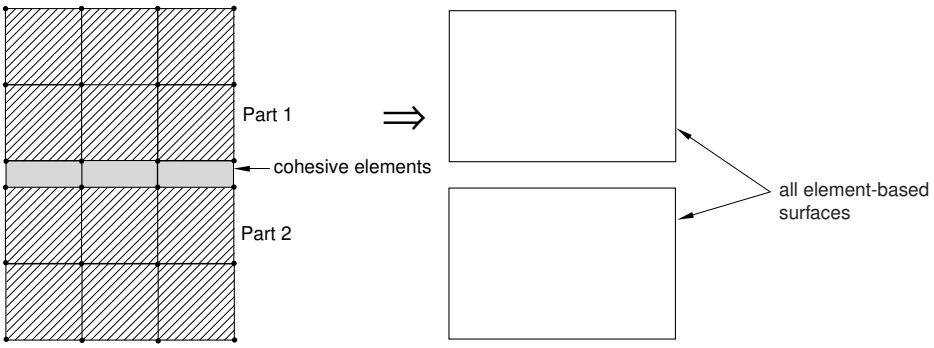


Figure 4: Default surface when cohesive elements share nodes with surrounding elements.

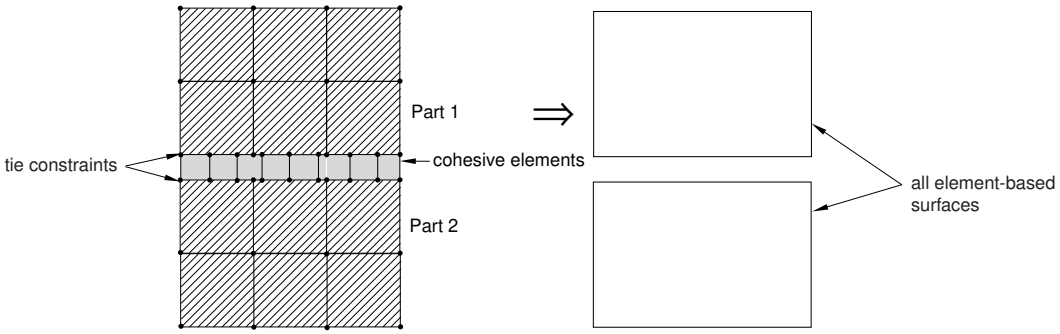


Figure 5: Default surface when cohesive elements are tied to the surrounding elements.

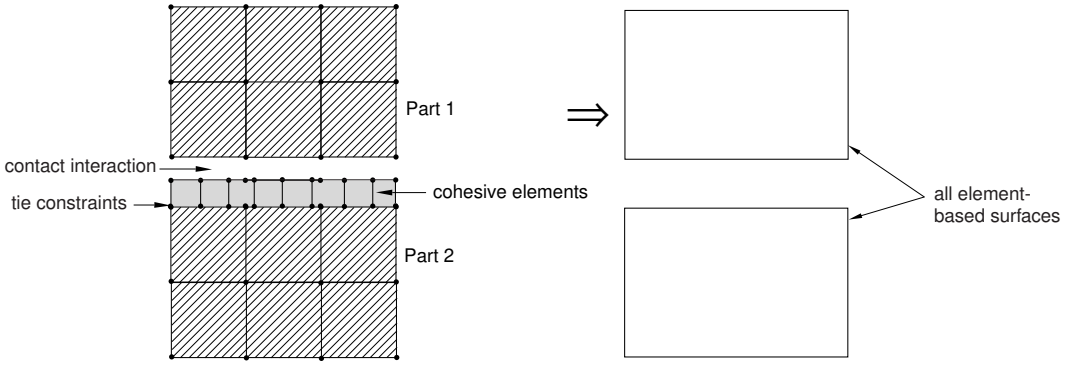


Figure 6: Default surface when cohesive elements are tied on one side and interact through contact on the other side.

Figure 7 shows the situation when the surfaces of the cohesive elements are also added to the default surface. Abaqus/Explicit generates a contact exclusion automatically so that the general contact algorithm avoids consideration of contact between the bottom surface of the cohesive elements and the top surface of Part 2 since these surfaces are tied together.

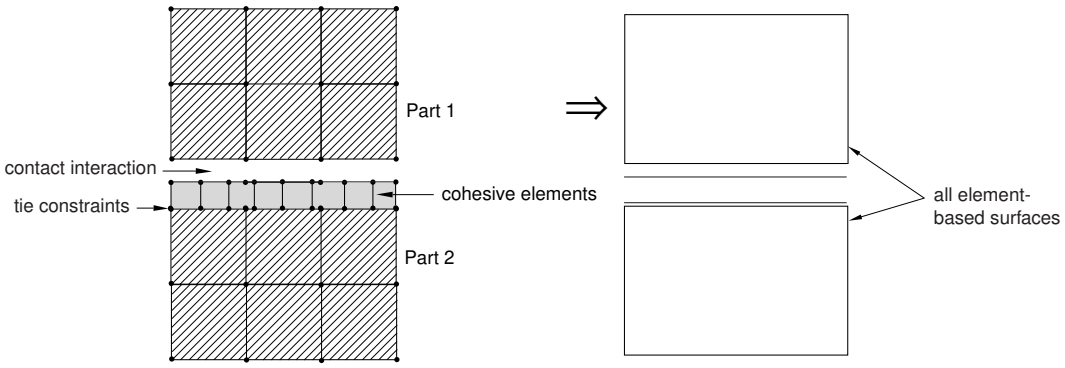


Figure 7: Top and bottom faces of the cohesive element along with the default surface when cohesive elements are tied on one side and interact through contact on the other side.

Input File Usage: Use the following options to add the top and bottom faces of the cohesive elements to the default general contact surface (the cohesive elements are included in the element set *COH_ELEMS*):

**SURFACE*, NAME=*DEFAULT_PLUS_COH*
,
COH_ELEMS,
**CONTACT*
**CONTACT INCLUSIONS*
DEFAULT_PLUS_COH,

Abaqus/CAE Usage: Any module except Sketch, Job, and Visualization: **Tools->Surface->Create**: **Name:** *default_plus_coh*: pick faces in viewport

Interaction module: **Create Interaction**: **General contact (Explicit)**: **Included surface pairs: Selected surface pairs: Edit**, select the surfaces in the columns on the left, and click the arrows in the middle to transfer them to the list of included pairs

Activating Contact Only When the Cohesive Elements Are Completely Degraded and Deleted

For general contact in Abaqus/Explicit, yet another approach for modeling contact between the surrounding structures involves activating contact only when the cohesive elements are completely degraded and deleted from the model (see *Maximum Degradation and Choice of Element Removal*). For this approach the cohesive elements must share nodes with the neighboring element and the general contact definition must include surfaces on the top and bottom faces of the cohesive elements, as shown in *Figure 8*. Since each surface face of the cohesive elements directly opposes a surface face of a neighboring element, the general contact algorithm does not consider these faces active while both parent elements are active. However, if the cohesive element fails, the opposing surface faces become active.

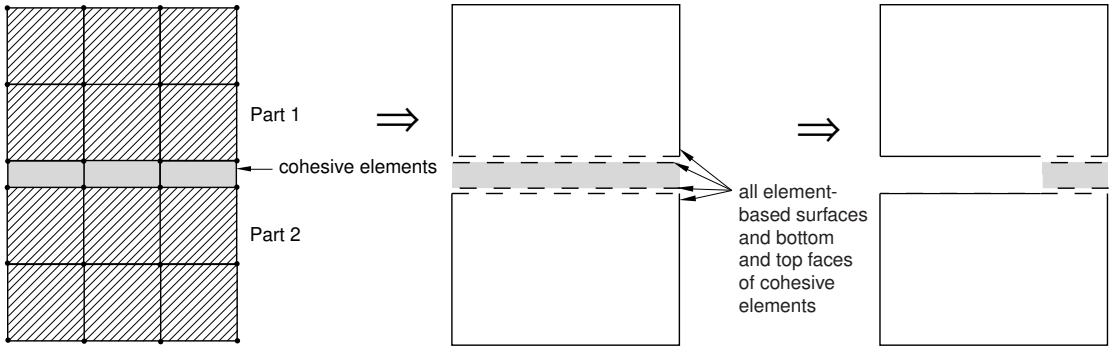


Figure 8: Surfaces that are involved in general contact when cohesive elements are included in the surface definition and erosion is used.

Input File Usage: Use the following options to include the top and bottom faces of the cohesive elements in the general contact definition (the cohesive elements are included in the element set *COH_ELEMS*):

**SURFACE*, NAME=*gc_surf*
,
COH_ELEMS,
**CONTACT*
**CONTACT INCLUSIONS*
gc_surf,

Abaqus/CAE Usage: Any module except Sketch, Job, and Visualization: **Tools->Surface->Create**: **Name:** *gc_surf*: pick faces in viewport

Interaction module: **Create Interaction**: **General contact (Explicit)**: **Included surface pairs: Selected surface pairs: Edit**, select the surfaces in the columns on the left, and click the arrows in the middle to transfer them to the list of included pairs

Using Cohesive Elements in Symmetric Models in Abaqus/Standard

Symmetry has long been exploited by engineers to simplify problem solving and to reduce model sizes. This is also a common practice in finite element analyses. The symmetry of structure and loading conditions allows you to model only a portion (half, quarter, etc.) of the actual structure. When crack propagation or delamination coincides with the plane of geometric symmetry, you can arrange one layer of cohesive elements along the symmetry plane and model half of the structure, as shown in *Figure 9*. You can choose any plane that is perpendicular to one of the global coordinate axes as the symmetry plane. Symmetric deformation of the cohesive elements is enforced automatically; you do not need to apply any symmetric constraints or boundary conditions.

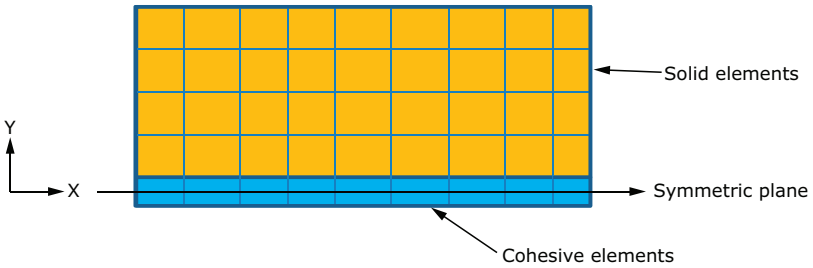


Figure 9: Cohesive elements in symmetric modeling.

If cohesive elements with pore pressure degrees of freedom are used to model tangential flow while concentrated fluid flow is used to define the fluid to be injected into the middle plane of cohesive elements, you can specify the flow magnitude as the same as the corresponding full model or half of the value. Specifying half of the value of the flow magnitude is recommended if the cohesive elements are connected to pipe elements.

Input File Usage: Use either of the following options to enforce symmetric modeling of cohesive elements without pore pressure degrees of freedom:

**COHESIVE SECTION*, SYMMETRIC

**COHESIVE SECTION*, HALF

Use one of the following sets of options to enforce symmetric modeling of cohesive elements with pore pressure degrees of freedom:

**COHESIVE SECTION*, SYMMETRIC
**CFLOW*
, , *flow magnitude same as corresponding full model*

or

**COHESIVE SECTION*, HALF
**CFLOW*
, , *half of the flow magnitude as the corresponding full model*

Abaqus/CAE Usage: Enforcing symmetric modeling of cohesive elements is not supported in Abaqus/CAE.

Stable Time Increment in Abaqus/Explicit

The stable time increment for a cohesive element in Abaqus/Explicit is equal to the time, Δt, required for a stress wave to travel across the constitutive thickness, T_c, of the cohesive layer:

$$\Delta t = \frac{T_c}{c},$$

where $c = \sqrt{\frac{E_c}{\rho_c}}$ is the wave speed and E_c and ρ_c represent the bulk stiffness and the density, respectively, of the adhesive material. In terms of the expression for the wave speed, the stable time increment can be written as

$$\Delta t = T_c \sqrt{\frac{\rho_c}{E_c}}.$$

For cases in which the constitutive response is defined in terms of traction versus separation, the slope of the traction versus separation relationship is $K_c = E_c/T_c$ and the density is specified as mass per unit area rather than per unit volume: $\overline{\rho}_c = \rho_c T_c$ (see *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description* for further details on this issue). Therefore, for traction versus separation the expression for the time increment becomes

$$\Delta t = \sqrt{\frac{\overline{\rho}_c}{K_c}}.$$

It is quite common that the time increment of cohesive elements will be significantly less than that of the other elements in the model, unless you take some action to alter one or more of the factors influencing the time increment. This requires some judgement on your part. The following discussions provide some recommendations for controlling the time increment for the different methods of defining the material response. However,

Abaqus/Standard may be preferable in some applications where it is necessary to model a thin, stiff cohesive layer without approximations.

Constitutive Response Defined in Terms of a Continuum or Uniaxial Stress-State Approach

For constitutive response defined in terms of a continuum or uniaxial stress-state approach, the ratio of the stable time increment of the cohesive elements to that of the other elements is given by

$$\frac{\Delta t_c}{\Delta t_e} = \left(\frac{T_c}{T_e}\right)\sqrt{\left(\frac{\rho_c}{\rho_e}\right)\left(\frac{E_e}{E_c}\right)},$$

where the subscripts “c” and “e” stand for the cohesive elements and the surrounding elements, respectively. The thickness of the cohesive layer is often smaller than a characteristic length of the other elements in the model, so the quantity (T_c/T_e) is often small. The quantity under the radical will depend on the materials involved. For an epoxy adhesive between steel components, the quantity under the radical is on the order of unity. The stable time increment of the cohesive element can be increased by artificially

- increasing the constitutive thickness, T_c;
- increasing the density, ρ_c;
- reducing the stiffness, E_c; or
- some combination of the above.

In many cases the most attractive option will be to increase the density, which is also referred to as mass scaling (*Mass Scaling*). However, if the thickness of the cohesive zone is very small, the mass scaling required to achieve a reasonable time increment may affect the results significantly. In such cases it may be necessary to artificially reduce the cohesive stiffness in addition to some mass scaling. This approach involves the use of a stiffness that is different from the measured stiffness of the interface; however, if the peak strength and the fracture energy remain unchanged, the global response will not be affected significantly in many cases.

Constitutive Response Defined in Terms of Traction Versus Separation

For constitutive response defined in terms of traction versus separation, the ratio of the stable time increment of the cohesive elements to that for the other elements is given by

$$\frac{\Delta t_c}{\Delta t_e} = \sqrt{\left(\frac{\bar{\rho}_c}{\bar{\rho}_e}\right)\left(\frac{K_e}{K_c}\right)},$$

where the subscripts “c” and “e” stand for the cohesive elements and the surrounding elements, respectively.

One way to ensure that the cohesive elements will have no adverse effect on the stable time increment is to choose material properties such that $\Delta t_c = \Delta t_e$, which implies

$$\frac{\bar{\rho}_c}{\bar{\rho}_e} = \frac{K_c}{K_e}.$$

This is accomplished if, for example, the cohesive element stiffness and density per unit area are chosen such that

$$K_c = \frac{E_c}{T_c} = \frac{1}{10}\frac{E_e}{T_e} = 0.1K_e,$$

$$\bar{\rho}_c = \rho_c T_c = \frac{1}{10}\rho_e T_e = 0.1\bar{\rho}_e,$$

where T_e represents the characteristic length of the neighboring non-cohesive elements. By choosing $K_c = 0.1K_e$, the stiffness in the cohesive layer relative to the surrounding elements will be similar to the default stiffness used by penalty contact in Abaqus/Explicit (relative to the equivalent one-dimensional stiffness of the surrounding

elements). This approach involves the use of a stiffness that is likely to be different from the measured stiffness of the interface; however, if the peak strength and the fracture energy remain unchanged, the global response will not be affected significantly in many cases.

Convergence Issues in Abaqus/Standard

In many problems cohesive elements are modeled as undergoing progressive damage leading to failure. The modeling of progressive damage involves softening in the material response, which is known to lead to convergence difficulties in an implicit solution procedure, such as in Abaqus/Standard. Convergence difficulties may also occur during unstable crack propagation, when the energy available is higher than the fracture toughness of the material. Several methods are available to help avoid these convergence problems.

Using Viscous Regularization

Abaqus/Standard provides a viscous regularization capability that helps in improving the convergence for these kinds of problems. This capability is discussed in detail in *Using Viscous Regularization with Cohesive Elements, Connector Elements, and Elements That Can Be Used with the Damage Evolution Models for Ductile Metals and Fiber-Reinforced Composites Including the Multiscale Model for Fiber-Reinforced Composites in Abaqus/Standard* and *Viscous Regularization in Abaqus/Standard*.

Using Automatic Stabilization

Another approach to help convergence behavior is the use of automatic stabilization (see *Static Stress Analysis* and *Solving Nonlinear Problems* for further details), which is useful when a problem is unstable due to local instabilities. Generally, if sufficient viscous regularization is used (as measured by the viscosity coefficient—see *Viscous Regularization in Abaqus/Standard* for further details), the use of the automatic stabilization technique is not necessary. In problems where a small amount or no viscous regularization is used, automatic stabilization will improve the convergence characteristics.

Using Nondefault Solution Controls

The use of nondefault solution controls (see *Commonly Used Control Parameters* and *Convergence Criteria for Nonlinear Problems* for further details) and activation of the line search technique (*Improving the Efficiency of the Solution by Using the Line Search Algorithm*) may be useful in improving the solution efficiency.

Defining the Cohesive Element's Initial Geometry

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Cohesive Elements*
- *Adhesive joints and bonded interfaces*

Overview

The initial geometry of a cohesive element is defined:

- by the nodal connectivity of the element and the position of these nodes;
- by the stack direction, which can be used to specify the top and the bottom faces of the cohesive element independent of the nodal connectivity; and
- by the magnitude of the initial constitutive thickness, which can either correspond to the geometric thickness implied by the nodal positions and stack direction or be specified directly.

Defining the Element Connectivity

The connectivity of a cohesive element is like that of a continuum element; however, it is useful to think of a cohesive element as being composed of two faces (a bottom and a top face) separated by the cohesive zone thickness. The element has nodes on its bottom face and corresponding nodes on its top face. Pore pressure cohesive elements include a third, middle face, which is used to model fluid flow within the element.

Three methods are available to define the element connectivity.

By Directly Defining the Element's Complete Connectivity

The complete connectivity of a cohesive element can be given directly (see *Defining Cohesive Elements*).

By Defining the Bottom-Face Element Connectivity and an Integer Offset

Alternatively, you can specify the connectivity of the bottom face plus a positive integer offset (see *Defining Cohesive Elements*) that will be used to determine the remaining cohesive element nodes.

Input File Usage: **ELEMENT*, OFFSET=*n*

Abaqus/CAE Usage: Element offsets are not supported in Abaqus/CAE.

Use with Displacement Cohesive Elements

The integer offset will be used to define node numbers of the top face of the cohesive element. Abaqus will automatically position the nodes of the top face to be coincident with those of the bottom face unless the nodes of the top face have already been assigned coordinates directly with a node definition (*Node Definition*).

Use with Pore Pressure-Displacement Cohesive Elements

When you define only the bottom face nodes, the integer offset will first be used to define the node numbers of the top face of the cohesive element, with the numbering of the top-face nodes offset from the bottom face node numbers. The integer offset will again be used to define the middle surface node numbers offset, with the

numbering of the middle-face nodes offset from the top face node numbers. Abaqus will automatically position the nodes of the top and middle faces to be coincident with those of the bottom face unless the nodes of the top face have already been assigned coordinates directly with a node definition (*Node Definition*).

By Defining the Bottom- and Top-Face Element Connectivities and an Integer Offset

For pore pressure cohesive elements, you also can specify the connectivity of the bottom and top faces plus a positive integer offset (see *Defining Cohesive Elements*) that will be used to determine the middle face cohesive element nodes.

When you define the bottom and top face nodes, the integer offset will be used to define the node numbers of the middle face, with the numbering of the middle-face nodes offset from the bottom face node numbers. Abaqus will automatically position the nodes of the middle face to be halfway between those of the bottom and top faces unless the nodes of the middle face have already been assigned coordinates directly with a node definition (*Node Definition*).

Input File Usage: **ELEMENT*, OFFSET=*n*

Abaqus/CAE Usage: Element offsets are not supported in Abaqus/CAE.

Specifying the out-of-Plane Thickness for Two-Dimensional Elements

For two-dimensional cohesive elements the out-of-plane thickness is required. You specify this additional information in the cohesive section definition; the default value is 1.0.

Input File Usage: **COHESIVE SECTION*
first data line
out-of-plane thickness

Abaqus/CAE Usage: Property module: cohesive section editor: toggle on **Out-of-plane thickness:** and specify the out-of-plane thickness

Specifying the Constitutive Thickness

You can specify the constitutive thickness of the cohesive element directly or allow Abaqus to compute it based on nodal coordinates such that the constitutive thickness is equal to the geometric thickness. The default behavior depends on the nature of the application.

If the geometric thickness of the cohesive element is very small compared to its surface dimensions, the thickness computed from the nodal coordinates may be inaccurate. In such cases you can specify a constant thickness directly when defining the section properties of these elements.

The characteristic element length of a cohesive element is equal to its constitutive thickness. The characteristic element length is often useful in defining the evolution of damage in materials (see *Mesh Dependency*).

When the Cohesive Element Response Is Based on a Continuum Approach

When the response of the cohesive elements is based on a continuum approach, by default the constitutive thickness of the element is computed by Abaqus based on the nodal coordinates. You can override this default by specifying the constitutive thickness directly.

Input File Usage: Use the following option to have Abaqus compute the thickness based on the nodal coordinates:

**COHESIVE SECTION*, RESPONSE=CONTINUUM,
THICKNESS=GEOMETRY (default)

Use the following option to specify the thickness directly:

**COHESIVE SECTION*, RESPONSE=CONTINUUM,
THICKNESS=SPECIFIED
thickness (1.0 by default)

Abaqus/CAE Usage: Property module: cohesive section editor: **Response**: **Continuum**: **Initial thickness**: **Use nodal coordinates**, **Specify**: *thickness*, or **Use analysis default**

When the Cohesive Element Response Is Based on a Traction-Separation Approach

When the response of the cohesive elements is based on a traction-separation approach, Abaqus assumes by default that the constitutive thickness is equal to one. This default value is motivated by the fact that the geometric thickness of cohesive elements is often equal to (or very close to) zero for the kinds of applications in which a traction-separation-based constitutive response is appropriate. This default choice ensures that nominal strains are equal to the relative separation displacements (see *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description* for further details). You can override this default by specifying another value or specifying that the constitutive thickness should be equal to the geometric thickness.

Input File Usage: Use the following option to specify the thickness directly:

**COHESIVE SECTION*, RESPONSE=TRACTION SEPARATION,
THICKNESS=SPECIFIED (default)
thickness (1.0 by default)

Use the following option to have Abaqus compute the thickness based on the nodal coordinates:

**COHESIVE SECTION*, RESPONSE=TRACTION SEPARATION,
THICKNESS=GEOMETRY

Abaqus/CAE Usage: Property module: cohesive section editor: **Response**: **Traction Separation**: **Initial thickness**: **Specify**: *thickness*, **Use analysis default**, or **Use nodal coordinates**

When the Cohesive Element Response Is Based on a Uniaxial Stress State

When the response of the cohesive elements is based on a uniaxial stress state, there is no default method for computing the constitutive thickness. You must indicate your choice of the method of determining the constitutive thickness.

Input File Usage: Use the following option to specify the thickness:

**COHESIVE SECTION*, RESPONSE=GASKET,
THICKNESS=SPECIFIED
thickness (1.0 by default)

Use the following option to have Abaqus compute the thickness based on the nodal coordinates:

*COHESIVE SECTION, RESPONSE=GASKET,
THICKNESS=GEOMETRY

Abaqus/CAE Usage: Property module: cohesive section editor: **Response**: **Gasket**: **Initial thickness**: **Specify**: *thickness* or **Use nodal coordinates**

Element Thickness Direction Definition

It is important to define the orientation of cohesive elements correctly, since the behavior of the elements is different in the thickness and in-plane directions. By default, the top and bottom faces of cohesive elements are as shown in *Figure 1* for three-dimensional cohesive elements and *Figure 2* for two-dimensional and axisymmetric cohesive elements. Options for overriding the default orientation of cohesive elements are discussed below along with an explanation of how the local thickness direction and in-plane direction vectors are established.

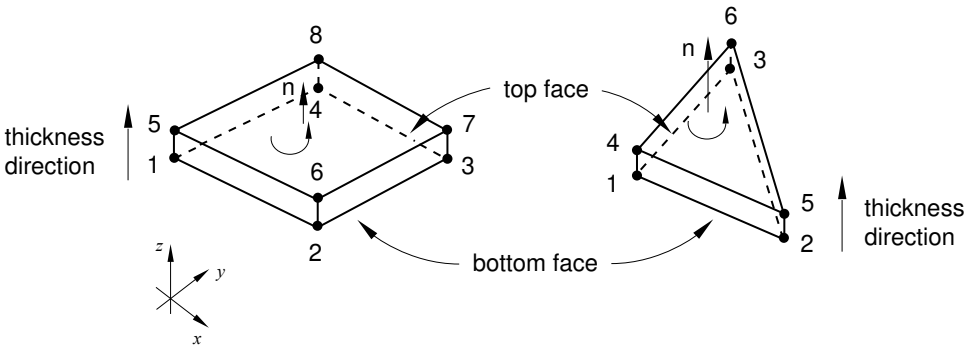


Figure 1: Default thickness direction for three-dimensional cohesive elements.

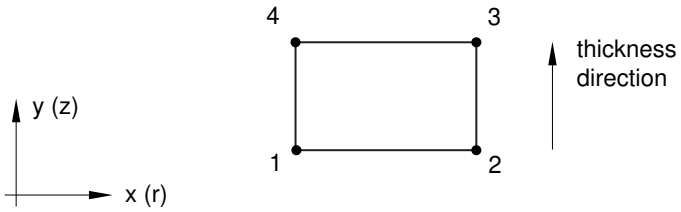
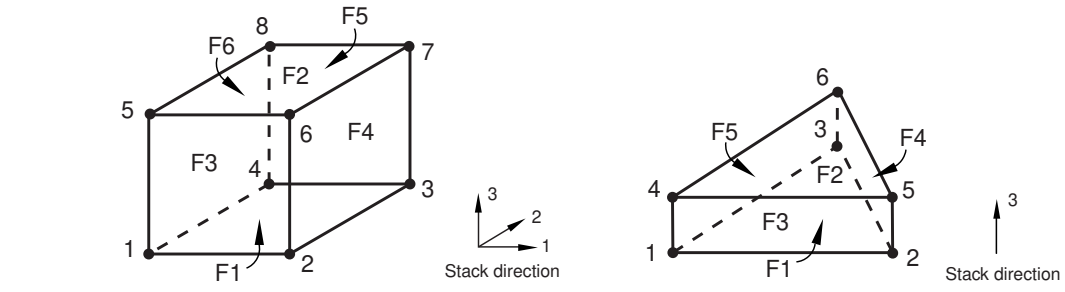


Figure 2: Default thickness direction for two-dimensional and axisymmetric cohesive elements.

Setting the Stack Direction Equal to an Isoparametric Direction

The “stack direction” refers to the isoparametric direction along which the top and bottom faces of a cohesive element are stacked. By default, the top and bottom faces are stacked along the third isoparametric direction in three-dimensional cohesive elements and along the second isoparametric direction in two-dimensional and axisymmetric cohesive elements. You can choose to stack the top and bottom faces along an alternate isoparametric direction for most element types (the COH3D6 element can have only the third isoparametric direction as the stack direction). The choice of the isoparametric direction depends on the element connectivity. For a mesh-independent specification, use an orientation-based method as described below. The isoparametric direction choices for three-dimensional cohesive elements are shown in *Figure 3*.



Stack direction = 1 from face 6 to face 4

Stack direction = 2 from face 3 to face 5

Stack direction = 3 from face 1 to face 2

Stack direction = 3 from face 1 to face 2

Figure 3: Stack directions for COH3D8 (left) and COH3D6 (right) elements.

Input File Usage: Use the following option to define the element top and bottom faces based on the element's isoparametric directions:

**COHESIVE SECTION*, STACK DIRECTION=*n*

Abaqus/CAE Usage: You cannot define the stack direction based on isoparametric directions in Abaqus/CAE. The stack direction will correspond to the default discussed above.

Setting the Stack Direction Based on a User-Defined Orientation

You can also control the orientation of the stack direction through a user-defined local orientation (*Orientations*). When you define an orientation for cohesive elements, you also specify an axis about which the local 1 and 2 material directions may be rotated. This axis also defines an approximate normal direction. The stack direction will be the element isoparametric direction that is closest to this approximate normal (see *Figure 4*).

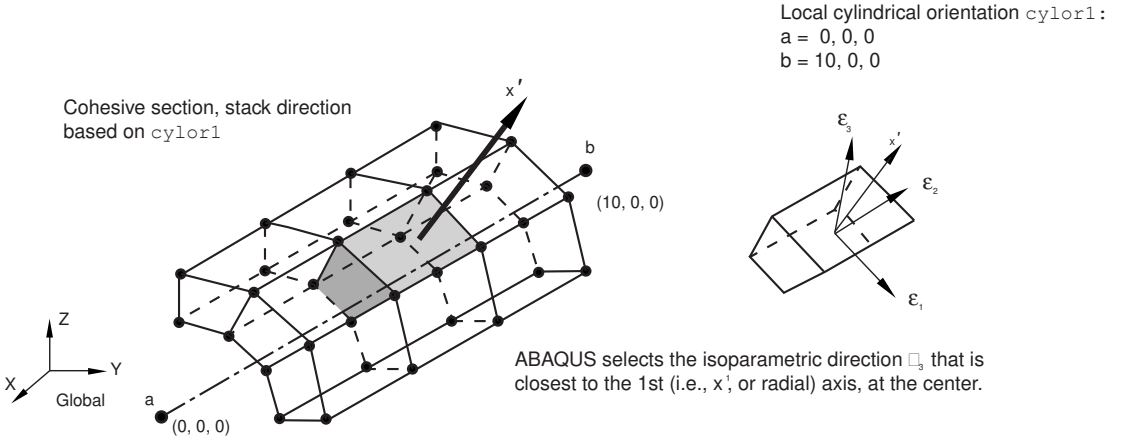


Figure 4: Example illustrating the use of a cylindrical system to define the stack direction.

Input File Usage: Use the following option to define the element thickness direction based on a user-defined orientation:

**COHESIVE SECTION*, STACK DIRECTION=ORIENTATION, ORIENTATION=*name*

Abaqus/CAE Usage: You cannot define the stack direction based on an orientation definition in Abaqus/CAE. The stack direction will correspond to the default discussed above.

Verifying the Stack Direction

The stack direction can be verified visually in Abaqus/CAE by using the stack direction query tool (see *Understanding the role of the Query toolset*). For three-dimensional elements Abaqus/CAE colors the top face brown and the bottom face purple. For two-dimensional and axisymmetric elements, arrows indicate the orientation of the element. In addition, Abaqus/CAE highlights any element faces and edges that have inconsistent orientations.

Alternatively, the material axes can be plotted in the Visualization module of Abaqus/CAE to verify that the 3-axis points in the desired normal direction for three-dimensional elements; and if the element is oriented improperly, one of the in-plane axes (either the 1- or 2-axis) will point in the normal direction. For two-dimensional and axisymmetric elements, the stack direction is consistent with the 2-axis material direction.

Thickness Direction Computation for Two-Dimensional and Axisymmetric Elements

To compute the thickness direction for two-dimensional and axisymmetric elements, Abaqus forms a midsurface by averaging the coordinates of the node pairs forming the bottom and top surfaces of the element. This midsurface passes through the integration points of the element, as shown in *Figure 5* for the default choice of the bottom and top surfaces. For each integration point Abaqus computes a tangent whose direction is defined by the sequence of nodes given on the bottom and top surfaces. The thickness direction is then obtained as the cross product of the out-of-plane and tangent directions.

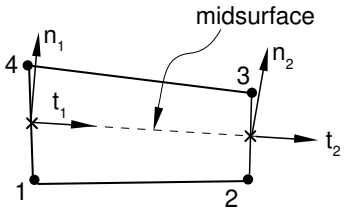


Figure 5: Thickness direction for a two-dimensional or axisymmetric element.

Thickness Direction Computation for Three-Dimensional Elements

To compute the thickness direction for three-dimensional elements, Abaqus forms a midsurface by averaging the coordinates of the node pairs forming the bottom and top surfaces of the element. This midsurface passes through the integration points of the element, as shown in *Figure 6* for the default choice of the bottom and top surfaces. Abaqus computes the thickness direction as the normal to the midsurface at each integration point; the positive direction is obtained with the right-hand rule going around the nodes of the element on the bottom or top surface.

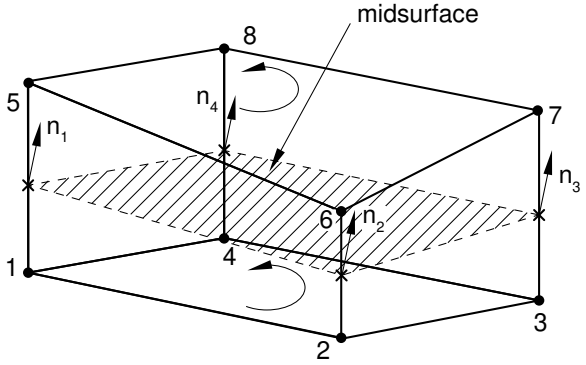


Figure 6: Thickness direction for a three-dimensional element.

Local Directions at Integration Points

Abaqus computes default local directions at each integration point. The local directions are used for output of all quantities that describe the current deformation state of a cohesive element. Details of local directions are discussed separately below for cohesive elements with two versus three local directions.

Local Directions for Two-Dimensional and Axisymmetric Cohesive Elements

The local 2-direction for two-dimensional and axisymmetric cohesive elements corresponds to the thickness direction, which is computed as discussed above in *Element Thickness Direction Definition*. The local 1-direction is defined such that the cross product between the local 1- and 2-directions gives the out-of-plane direction (see *Figure 7*). You cannot modify either local direction for these elements for a given stack orientation. Transverse shear behavior is defined in the 1–2 plane for these elements.

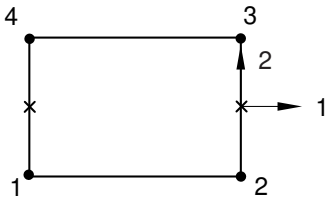


Figure 7: Local directions for two-dimensional and axisymmetric cohesive elements.

Local Directions for Three-Dimensional Cohesive Elements

The local 3-direction for three-dimensional cohesive elements corresponds to the thickness direction, which is computed as discussed above in *Element Thickness Direction Definition* and cannot be modified for a given stack orientation. The local 1- and 2-directions are normal to the thickness direction and, by default, are defined by the standard Abaqus convention for local directions on surfaces (*Conventions*). The default local directions for a three-dimensional cohesive element are shown in *Figure 8*.

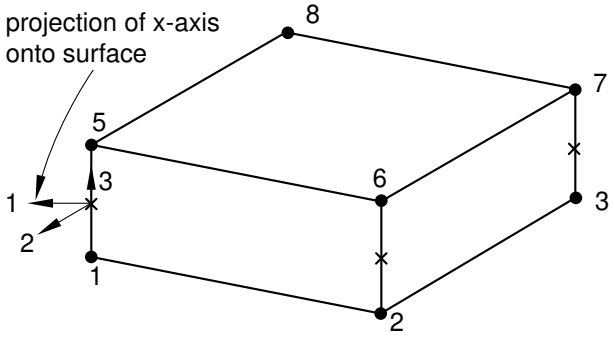


Figure 8: Local directions for three-dimensional cohesive elements.

Transverse shear behavior is defined in the local 1–3 and 2–3 planes for these elements. You can modify the local 1- and 2-directions for three-dimensional cohesive elements in the plane normal to the thickness direction by using a local orientation definition (*Orientations*).

Input File Usage: **COHESIVE SECTION*, ELSET=*name*, ORIENTATION=*name*

Abaqus/CAE Usage: Property module: **Assign->Material Orientation**: select region: select orientation

Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Cohesive Elements*
- *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*
- *About Progressive Damage and Failure*
- **COHESIVE SECTION*
- **TRANSVERSE SHEAR STIFFNESS*
- *Adhesive joints and bonded interfaces*

Overview

The features described in this section are used to model cohesive elements using a continuum approach, which assumes that the cohesive zone contains material of finite thickness that can be modeled using the conventional material models in Abaqus. If the cohesive zone is very thin and for all practical purposes may be considered to be of zero thickness, the constitutive response is commonly described in terms of a traction-separation law; this alternative approach is discussed in *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*.

The constitutive response of cohesive elements modeled as a continuum:

- can be defined in terms of macroscopic material properties such as stiffness and strength using conventional material models;
- can be specified in terms of either a built-in material model or a user-defined material model;
- can include the effects of material damage and failure in Abaqus/Explicit; and
- can also include the effects of material damage and failure in a low-cycle fatigue analysis in Abaqus/Standard.

Behavior of Cohesive Elements with Conventional Material Models

The implementation of the conventional material models (including user-defined models) in Abaqus for cohesive elements is based on certain assumptions regarding the state of the deformation in the cohesive layer. Two different classes of problems are considered: modeling of an adhesive layer of finite thickness and modeling of gaskets.

Modeling of damage with cohesive elements for these classes of problems can be carried out in both Abaqus/Standard and Abaqus/Explicit (see *About Progressive Damage and Failure* for details regarding the damage models). You may need to alter the damage model for an adhesive material to account for the fact that the failure of an adhesive bond may occur at the interface between the adhesive and the adherend rather than within the adhesive material.

When used with conventional material models in Abaqus, cohesive elements use true stress and strain measures. When used with a material model that is based on a traction-separation description (see *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description* for details on this approach), cohesive elements use nominal stress and strain measures.

The frequency characteristics of cohesive elements are accounted for by the algorithms to automatically choose the time increment in Abaqus/Explicit (*Explicit Dynamic Analysis*). In many applications involving adhesives or gaskets cohesive elements may be quite thin compared to the other elements, which tends to decrease the stable time increment. See *Stable Time Increment in Abaqus/Explicit* for further discussion on this topic, including suggestions on how to avoid significant reductions in the stable time increment when using cohesive elements.

Modeling of an Adhesive Layer of Finite Thickness

For adhesive layers with finite thickness it is assumed that the cohesive layer is subjected to only one direct component of strain, which is the through-thickness strain, and to two transverse shear strain components (one transverse shear strain component for two-dimensional problems). The other two direct components of the strain (the direct membrane strains) and the in-plane (membrane) shear strain are assumed to be zero for the constitutive calculations. More specifically, the through-thickness and the transverse shear strains are computed from the element kinematics. However, the membrane strains are not computed based on the element kinematics; they are simply assumed to be zero for the constitutive calculations. These assumptions are appropriate in situations where a relatively thin and compliant layer of adhesive bonds two relatively rigid (compared to the adhesive) parts. The above kinematic assumptions are approximately correct everywhere inside the cohesive layer except around its outer edges.

An additional linear elastic transverse shear behavior can be defined to provide more stability to cohesive elements, particularly after damage has occurred. The transverse shear behavior is assumed to be independent of the regular material response and does not undergo any damage.

Input File Usage: Use the following options (the second option is needed only to define uncoupled transverse shear response):

**COHESIVE SECTION*, RESPONSE=CONTINUUM
**TRANSVERSE SHEAR STIFFNESS*

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Cohesive** as the section **Type**: **Response**: **Continuum**

Transverse shear behavior is not supported in Abaqus/CAE for cohesive sections.

Modeling of Gaskets and/or Small Adhesive Patches

The modeling of gaskets and/or small adhesive patches involves situations where there are no lateral constraints on the cohesive layer. Hence, the layers are free to expand in the lateral direction in a stress-free manner. Application areas include individual spot welds and gaskets. The constitutive calculations assume only one direct stress component, which is the through-thickness normal stress. All other stress components, including the transverse shear stress components, are assumed to be zero.

The gasket modeling capability that is offered with this option has some advantages compared to the family of gasket elements in Abaqus/Standard. The cohesive elements are fully nonlinear (the element kinematics properly account for finite strains as well as finite rotations), can contribute mass and damping in a dynamic analysis, and are available in Abaqus/Explicit. The gasket response modeled in the above manner is similar to modeling using the special-purpose gasket elements in Abaqus/Standard with thickness-direction behavior only (see *Including Gasket Elements in a Model*).

Uncoupled, linear-elastic transverse shear behavior, if desired, can be defined. The transverse shear behavior may either define the response of the gasket and/or adhesive patch or provide stability after damage has occurred in the response in the thickness direction. There is no damage associated with the transverse shear response.

Input File Usage: Use the following options (the second option is needed only to define uncoupled transverse shear response):

**COHESIVE SECTION*, RESPONSE=GASKET
**TRANSVERSE SHEAR STIFFNESS*

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Cohesive** as the section **Type**: **Response**: **Gasket**

Transverse shear behavior is not supported in Abaqus/CAE for cohesive sections.

Output

All standard output variables in Abaqus (*Abaqus/Standard Output Variable Identifiers* and *Abaqus/Explicit Output Variable Identifiers*) are available for cohesive elements that are used with conventional material models. The stresses due to the additional transverse shear response are reported separately using the output variables TSHR13 and (in three dimensions) TSHR23. These stresses are not added to the usual material point stresses reported using the output variable S.

Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Cohesive Elements*
- *Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach*
- **COHESIVE SECTION*
- **DAMAGE EVOLUTION*
- **DAMAGE INITIATION*
- *Defining damage*
- *Adhesive joints and bonded interfaces*

Overview

The features described in this section are primarily intended for bonded interfaces where the interface thickness is negligibly small. In such cases it might be straightforward to define the constitutive response of the cohesive layer directly in terms of traction versus separation.

If the interface adhesive layer has a finite thickness and macroscopic properties (such as stiffness and strength) of the adhesive material are available, it might be more appropriate to model the response using conventional material models. The former approach is discussed in this section, while the latter approach is discussed in *Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach*.

Cohesive behavior defined directly in terms of a traction-separation law:

- can be used to model the delamination at interfaces in composites directly in terms of traction versus separation;
- allows specification of material data such as the fracture energy as a function of the ratio of normal to shear deformation (mode mix) at the interface;
- assumes a linear elastic traction-separation law prior to damage;
- can be used in combination with linear viscoelasticity in Abaqus/Explicit (*Defining Viscoelastic Behavior for Traction-Separation Elasticity in Abaqus/Explicit*) and rate-dependent damage initiation and evolution to describe delamination behavior that depends on the rate of separation;
- assumes that failure of the elements is characterized by progressive degradation of the material stiffness, which is driven by a damage process;
- allows multiple damage mechanisms; and
- can be used with user subroutine *UMAT* in Abaqus/Standard or *VUMAT* in Abaqus/Explicit to specify user-defined traction-separation laws.

Defining Constitutive Response in Terms of Traction-Separation Laws

To define the constitutive response of the cohesive element directly in terms of traction versus separation, you choose a traction-separation response when defining the section behavior of the cohesive elements.

Input File Usage: **COHESIVE SECTION*, RESPONSE=TRACTION SEPARATION

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Cohesive** as the section **Type**: **Response**: **Traction Separation**

Linear Elastic Traction-Separation Behavior

The available traction-separation model in Abaqus assumes initially linear elastic behavior (see *Defining Elasticity in Terms of Tractions and Separations for Cohesive Elements*) followed by the initiation and evolution of damage. The elastic behavior is written in terms of an elastic constitutive matrix that relates the nominal stresses to the nominal strains across the interface. The nominal stresses are the force components divided by the original area at each integration point. The nominal strains are the separations divided by the original thickness at each integration point with some averaging applied to the transverse shear components. The default value of the original constitutive thickness is 1.0 if traction-separation response is specified, which ensures that the nominal strain is equal to the separation (that is, relative displacements of the top and bottom faces). The constitutive thickness used for traction-separation response is typically different from the geometric thickness (which is typically close or equal to zero). See *Specifying the Constitutive Thickness* for a discussion on how to modify the constitutive thickness.

The nominal traction stress vector, $\mathbf{t}$, consists of three components (two components in two-dimensional problems): t_n, t_s, and (in three-dimensional problems) t_t, which represent the normal (along the local 3-direction in three dimensions and along the local 2-direction in two dimensions) and the two shear tractions (along the local 1- and 2-directions in three dimensions and along the local 1-direction in two dimensions), respectively. The corresponding separations are denoted by δ_n, δ_s, and δ_t. Denoting by T_o the original thickness of the cohesive element, the nominal strains can be defined as

$$\varepsilon_n = \frac{\delta_n}{T_o}, \varepsilon_s = \frac{\delta_s}{T_o}, \varepsilon_t = \frac{\delta_t}{T_o}.$$

The elastic behavior can then be written as

$$\mathbf{t} = \begin{Bmatrix} t_n \\ t_s \\ t_t \end{Bmatrix} = \begin{bmatrix} E_{nn} & E_{ns} & E_{nt} \\ E_{ns} & E_{ss} & E_{st} \\ E_{nt} & E_{st} & E_{tt} \end{bmatrix} \begin{Bmatrix} \varepsilon_n \\ \varepsilon_s \\ \varepsilon_t \end{Bmatrix} = \mathbf{E}\boldsymbol{\varepsilon}.$$

The elasticity matrix provides fully coupled behavior between all components of the traction vector and separation vector and can depend on temperature and field variables. Set the off-diagonal terms in the elasticity matrix to zero if uncoupled behavior between the normal and shear components is required.

Optionally, for the uncoupled traction behavior a compression factor can be specified; this ensures that the compressive stiffness is equal to the specified factor times the tensile stiffness, E_{nn}. This factor affects only the traction response for separation in the normal direction; the shear behavior is not affected.

Input File Usage: Use the following option to define uncoupled traction-separation behavior:

**ELASTIC*, TYPE=TRACTION

Use the following option to define uncoupled traction-separation behavior with a compression factor:

**ELASTIC*, TYPE=TRACTION, COMPRESSION FACTOR=*f*

Use the following option to define coupled traction-separation behavior:

**ELASTIC*, TYPE=COUPLED TRACTION

Abaqus/CAE Usage: Use the following option to define uncoupled traction-separation behavior:

Property module: material editor: **Mechanical->Elasticity->Elastic**: **Type**: **Traction**

Use the following option to define coupled traction-separation behavior:

Property module: material editor: **Mechanical->Elasticity->Elastic**: **Type**: **Coupled Traction**

Specifying a compression factor for uncoupled traction-separation behavior is not supported in Abaqus/CAE.

Interpretation of Material Properties

The material parameters, such as the interfacial elastic stiffness, for a traction-separation model can be better understood by studying the equation that represents the displacement of a truss of length L, elastic stiffness E, and original area A, due to an axial load P:

$$\delta = \frac{PL}{AE}.$$

This equation can be rewritten as

$$\delta = \frac{S}{K},$$

where $S = P/A$ is the nominal stress and $K = E/L$ is the stiffness that relates the nominal stress to the displacement. Likewise, the total mass of the truss, assuming a density ρ, is given by

$$M = \rho AL = \overline{\rho}\, A.$$

The above equations suggest that the actual length L can be replaced with 1.0 (to ensure that the strain is the same as the displacement) if the stiffness and the density are appropriately reinterpreted. In particular, the stiffness is $K = (E/L)$ and the density is $\overline{\rho} = (\rho L)$, where the true length of the truss is used in these equations. The density represents mass per unit area instead of mass per unit volume.

These ideas can be carried over to a cohesive layer of initial thickness T_c. If the adhesive material has stiffness E_a and density ρ_c, the stiffness of the interface (relating the nominal traction to the nominal strain) is given by $E_c = (E_a/T_c)\, T_o$ and the density of the interface is given by $\overline{\rho_c} = (\rho_c T_c)$. As discussed earlier, the default choice of the constitutive thickness T_o for modeling the response in terms of traction versus separation is 1.0 regardless of the actual thickness of the cohesive layer. With this choice, the nominal strains are equal to the corresponding separations. When the constitutive thickness of the cohesive layer is "artificially" set to 1.0, ideally you should specify E_c and $\overline{\rho_c}$ (if needed) as the material stiffness and density, respectively, as calculated with the true thickness of the cohesive layer.

The above formulas provide a recipe for estimating the parameters required for modeling the traction-separation behavior of an interface in terms of the material properties of the bulk adhesive material. As the thickness of the interface layer tends to zero, the above equations imply that the stiffness, E_c, tends to infinity and the density, $\overline{\rho_c}$, tends to zero. This stiffness is often chosen as a penalty parameter. A very large penalty stiffness is detrimental to the stable time increment in Abaqus/Explicit and might result in ill-conditioning of the element operator in Abaqus/Standard. Recommendations for the choice of the stiffness and density of an interface for an Abaqus/Explicit analysis such that the stable time increment is not adversely affected are provided in *Stable Time Increment in Abaqus/Explicit*.

Modeling Viscoelastic Traction-Separation Behavior in Abaqus/Explicit

Time domain viscoelasticity can be used in Abaqus/Explicit to model rate-dependent behavior of cohesive elements with traction-separation elasticity. The evolution equation for the normal and two shear nominal tractions take the form:

$$t_n(t) = t_n^0(t) + \int_0^t \dot{k}_R(s)\, t_n^0(t-s)\, ds,$$

$$t_s\left(t\right)=t_s^0\left(t\right)+\int_0^t \dot{g}_R\left(s\right)t_s^0\left(t-s\right)\,ds,$$
$$t_t\left(t\right)=t_t^0\left(t\right)+\int_0^t \dot{g}_R\left(s\right)t_t^0\left(t-s\right)\,ds,$$

where $t_n^0\left(t\right)$, $t_s^0\left(t\right)$, and $t_t^0\left(t\right)$ are the instantaneous nominal tractions at time t in the normal and the two local shear directions, respectively. The functions $g_R\left(t\right)$ and $k_R\left(t\right)$ represent the dimensionless shear and normal relaxation moduli, respectively. See *Defining Viscoelastic Behavior for Traction-Separation Elasticity in Abaqus/Explicit* for additional details and usage information.

You can also combine time domain viscoelasticity with the models for rate-dependent progressive damage and failure described in the next sections. This combination allows modeling rate-dependent behavior both during the initial elastic response (prior to damage initiation), as well as during damage progression.

Damage Modeling

Both Abaqus/Standard and Abaqus/Explicit allow modeling of progressive damage and failure in cohesive layers whose response is defined in terms of traction-separation. By comparison, only Abaqus/Explicit allows modeling of progressive damage and failure for cohesive elements modeled with conventional materials (*Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach*). Damage of the traction-separation response is defined within the same general framework used for conventional materials (see *About Progressive Damage and Failure*). This general framework allows the combination of several damage mechanisms acting simultaneously on the same material. Each failure mechanism consists of three ingredients: a damage initiation criterion, a damage evolution law, and a choice of element removal (or deletion) on reaching a completely damaged state. While this general framework is the same for traction-separation response and conventional materials, many details of how the various ingredients are defined are different. Therefore, the details of damage modeling for traction-separation response are presented below.

The initial response of the cohesive element is assumed to be linear as discussed above. However, once a damage initiation criterion is met, material damage can occur according to a user-defined damage evolution law. *Figure 1* shows a typical traction-separation response with a failure mechanism. If the damage initiation criterion is specified without a corresponding damage evolution model, Abaqus will evaluate the damage initiation criterion for output purposes only; there is no effect on the response of the cohesive element (that is, no damage will occur). The cohesive layer does not undergo damage under pure compression.

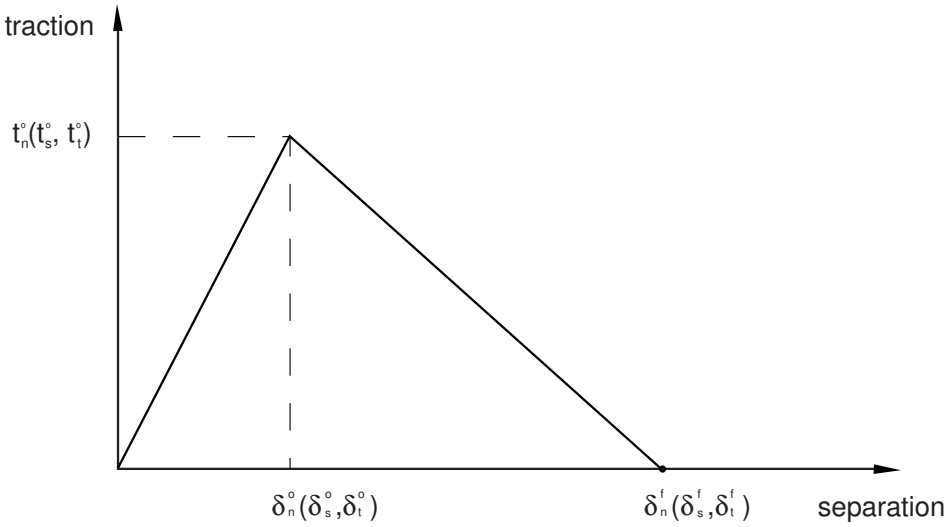


Figure 1: Typical traction-separation response.

Damage Initiation

As the name implies, damage initiation refers to the beginning of degradation of the response of a material point. The process of degradation begins when the stresses and strains satisfy certain damage initiation criteria that you specify. Several damage initiation criteria are available and are discussed below. Each damage initiation criterion also has an output variable associated with it to indicate whether the criterion is met. A value of 1 or higher indicates that the initiation criterion has been met (see *Output* for further details). Damage initiation criteria that do not have an associated evolution law affect only output. Thus, you can use these criteria to evaluate the propensity of the material to undergo damage without actually modeling the damage process (that is, without actually specifying damage evolution).

In the discussion below, t_n^o, t_s^o, and t_t^o represent the peak values of the nominal stress when the deformation is either purely normal to the interface or purely in the first or the second shear direction, respectively. Likewise, ε_n^o, ε_s^o, and ε_t^o represent the peak values of the nominal strain when the deformation is either purely normal to the interface or purely in the first or the second shear direction, respectively. With the initial constitutive thickness $T_o = 1$, the nominal strain components are equal to the respective components of the relative displacement—δ_n, δ_s, and δ_t—between the top and bottom of the cohesive layer. The symbol $\langle\rangle$ used in the discussion below represents the Macaulay bracket with the usual interpretation. The Macaulay brackets are used to signify that a pure compressive deformation or stress state does not initiate damage.

Maximum Nominal Stress Criterion

Damage is assumed to initiate when the maximum nominal stress ratio (as defined in the expression below) reaches a value of one. This criterion can be represented as

$$\max\left\{\frac{\langle t_n\rangle}{t_n^o}, \frac{t_s}{t_s^o}, \frac{t_t}{t_t^o}\right\} = 1.$$

Input File Usage: **DAMAGE INITIATION*, CRITERION=MAXS

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Maxs Damage**

Maximum Nominal Strain Criterion

Damage is assumed to initiate when the maximum nominal strain ratio (as defined in the expression below) reaches a value of one. This criterion can be represented as

$$\max\left\{\frac{\langle \varepsilon_n\rangle}{\varepsilon_n^o}, \frac{\varepsilon_s}{\varepsilon_s^o}, \frac{\varepsilon_t}{\varepsilon_t^o}\right\} = 1.$$

Input File Usage: **DAMAGE INITIATION*, CRITERION=MAXE

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Maxe Damage**

Quadratic Nominal Stress Criterion

Damage is assumed to initiate when a quadratic interaction function involving the nominal stress ratios (as defined in the expression below) reaches a value of one. This criterion can be represented as

$$\left\{\frac{\langle t_n\rangle}{t_n^o}\right\}^2 + \left\{\frac{t_s}{t_s^o}\right\}^2 + \left\{\frac{t_t}{t_t^o}\right\}^2 = 1.$$

Input File Usage: **DAMAGE INITIATION*, CRITERION=QUADS

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Quads Damage**

Quadratic Nominal Strain Criterion

Damage is assumed to initiate when a quadratic interaction function involving the nominal strain ratios (as defined in the expression below) reaches a value of one. This criterion can be represented as

$$\left\{\frac{\langle \varepsilon_n \rangle}{\varepsilon_n^o}\right\}^2 + \left\{\frac{\varepsilon_s}{\varepsilon_s^o}\right\}^2 + \left\{\frac{\varepsilon_t}{\varepsilon_t^o}\right\}^2 = 1.$$

Input File Usage: **DAMAGE INITIATION*, CRITERION=QUADE

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Quade Damage**

Rate-Dependent Damage Initiation in Abaqus/Explicit

In Abaqus/Explicit you can define the damage initiation criteria as a function of the effective rate of separation. You can combine this definition with rate-dependent damage evolution to describe the rate-dependent delamination behavior for a cohesive interface.

Input File Usage: Use the following option to specify rate-dependent damage initiation:

**DAMAGE INITIATION*, RATE DEPENDENT

Abaqus/CAE Usage: Rate-dependent damage initiation is not supported in Abaqus/CAE.

Damage Evolution

The damage evolution law describes the rate at which the material stiffness is degraded once the corresponding initiation criterion is reached. The general framework for describing the evolution of damage in bulk materials (as opposed to interfaces modeled using cohesive elements) is described in *Damage Evolution and Element Removal for Ductile Metals*. Conceptually, similar ideas apply for describing damage evolution in cohesive elements with a constitutive response that is described in terms of traction versus separation; however, many details are different.

A scalar damage variable, D, represents the overall damage in the material and captures the combined effects of all the active mechanisms. It initially has a value of 0. If damage evolution is modeled, D monotonically evolves from 0 to 1 on further loading after the initiation of damage. The stress components of the traction-separation model are affected by the damage according to

$$t_n = \begin{cases} (1-D)\,\bar{t}_n, & \bar{t}_n \geq 0 \\ \bar{t}_n, & \text{otherwise (no damage to compressive stiffness);} \end{cases}$$

$$t_s = (1-D)\,\bar{t}_s,$$

$$t_t = (1-D)\,\bar{t}_t,$$

where $\bar{t}_n$, $\bar{t}_s$ and $\bar{t}_t$ are the stress components predicted by the elastic traction-separation behavior for the current strains without damage.

To describe the evolution of damage under a combination of normal and shear deformation across the interface, it is useful to introduce an effective displacement (Camanho and Davila, 2002) defined as

$$\delta_m = \sqrt{\langle \delta_n \rangle^2 + \delta_s^2 + \delta_t^2}.$$

The equivalent nominal strain can be defined as

$$\varepsilon_m = \frac{\delta_m}{T_0} = \sqrt{\langle \varepsilon_n \rangle^2 + \varepsilon_s^2 + \varepsilon_t^2}.$$

Mixed-Mode Definition

The mode mix of the deformation fields in the cohesive zone quantify the relative proportions of normal and shear deformation. Abaqus uses three measures of mode mix, two that are based on energies and one that is based on tractions. You can choose one of these measures when you specify the mode dependence of the damage evolution process. Denoting by G_n, G_s, and G_t the work done by the tractions and their conjugate relative displacements in the normal, first, and second shear directions, respectively, and defining $G_T = G_n + G_s + G_t$, the mode mix definitions based on energies are as follows:

$$m_1 = \frac{G_n}{G_T},$$

$$m_2 = \frac{G_s}{G_T},$$

$$m_3 = \frac{G_t}{G_T}.$$

Clearly, only two of the three quantities defined above are independent. It is also useful to define the quantity $G_S = G_s + G_t$ to denote the portion of the total work done by the shear traction and the corresponding relative displacement components. As discussed later, Abaqus requires that you specify material properties related to damage evolution as functions of $m_2 + m_3 \ \ (= G_S/G_T)$ (or, equivalently, $1 - m_1$) and $m_3/(m_2 + m_3) \ \ (= G_t/G_S)$.

Abaqus computes the energy quantities described above either based on the current state of deformation (nonaccumulative measure of energy) or based on the deformation history (accumulative measure of energy) at an integration point. The former approach is useful in mixed-mode simulations where the primary energy dissipation mechanism is associated with the creation of new surfaces due to failure in the cohesive zone. Typically, such problems are adequately described utilizing the methods of linear elastic fracture mechanics. The latter approach provides an alternate way of defining the mode mix and might be useful in situations where other significant dissipation mechanisms also govern the overall structural response.

The corresponding definitions of the mode mix based on traction components are given by

$$\phi_1 = \left(\frac{2}{\pi}\right) \tan^{-1}\left(\frac{\tau}{\langle t_n \rangle}\right),$$

$$\phi_2 = \left(\frac{2}{\pi}\right) \tan^{-1}\left(\left|\frac{t_t}{t_s}\right|\right),$$

where $\tau = \sqrt{t_s^2 + t_t^2}$ is a measure of the effective shear traction. The angular measures used in the above definition (before they are normalized by the factor $2/\pi$) are illustrated in *Figure 2*.

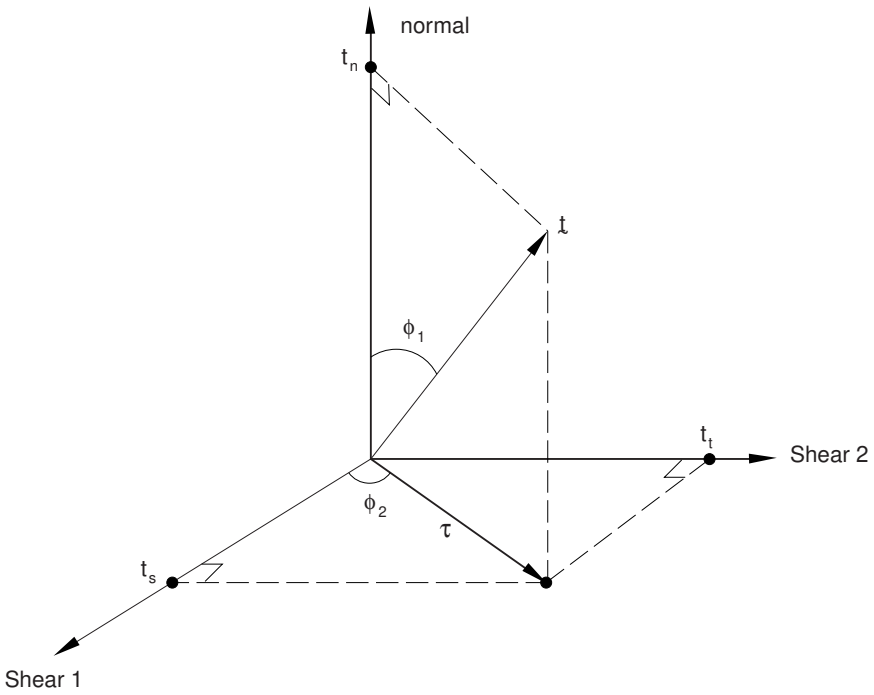


Figure 2: Mode mix measures based on traction.

Input File Usage: Use the following option to use the mode mix definition based on nonaccumulated energies:

**DAMAGE EVOLUTION*, MODE MIX RATIO=ENERGY

Use the following option to use the mode mix definition based on accumulated energies:

**DAMAGE EVOLUTION*, MODE MIX RATIO=ACCUMULATED ENERGY

Use the following option to use the mode mix definition based on tractions:

**DAMAGE EVOLUTION*, MODE MIX RATIO=TRACTION

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Quade Damage**, **Maxe Damage**, **Quads Damage**, or **Maxs Damage**: **Suboptions->Damage Evolution**: **Mode mix ratio:Energy** or **Traction**

Specifying a mode mix definition based on accumulated energies is not supported in Abaqus/CAE.

Comparison of Mixed-Mode Definitions

The mode mix ratios defined in terms of the different energy quantities and tractions can be quite different in general. The following examples illustrate this point. In terms of energies a deformation in the purely normal direction is one for which $G_n \neq 0$ and $G_s = G_t = 0$, irrespective of the values of the normal and the shear tractions. In particular, for a material with coupled traction-separation behavior both the normal and shear tractions might be nonzero for a deformation in the purely normal direction. For this case the definition of mode mix

based on energies would indicate a purely normal deformation, while the definition based on tractions would suggest a mix of both normal and shear deformation.

When the mode mix is defined based on accumulated energies, an artificial path-dependence might be introduced in the mixed-mode behavior that might not be consistent, for example, with predictions that are based on linear elastic fracture mechanics. Therefore, if an interface is first loaded purely in the normal deformation mode, unloaded, and subsequently loaded in a purely shear deformation mode, the mode mix ratios based on accumulated energies at the end of the above deformation path evaluate to (assuming the shear deformation to be in the local-1 direction only) $G_n \neq 0$ and $G_s \neq 0$. On the other hand, the mode mix ratios based on nonaccumulated energies evaluate to $G_n = 0$ and $G_s \neq 0$ at the end of the above deformation path.

Damage Evolution Definition

There are two components to the definition of the evolution of damage. The first component involves specifying either the effective displacement at complete failure, δ_m^f, relative to the effective displacement at the initiation of damage, δ_m^o; or the energy dissipated due to failure, G^C (see *Figure 3*).

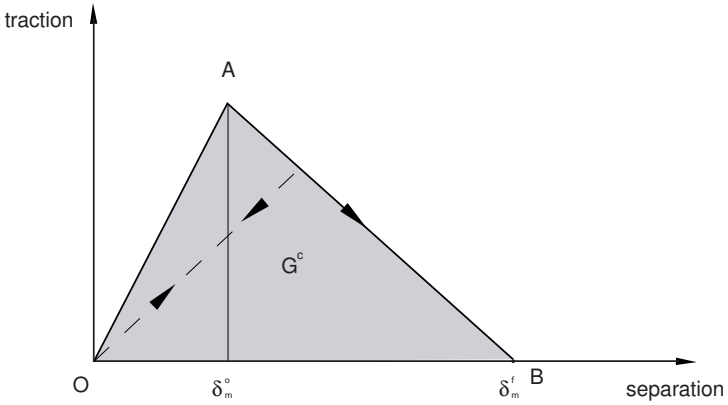


Figure 3: Linear damage evolution.

The second component to the definition of damage evolution is the specification of the nature of the evolution of the damage variable, D, between initiation of damage and final failure. This can be done by either defining linear or exponential softening laws or specifying D directly as a tabular function of the effective displacement relative to the effective displacement at damage initiation. The material data described above will in general be functions of the mode mix, temperature, and field variables.

Figure 4 is a schematic representation of the dependence of damage initiation and evolution on the mode mix, for a traction-separation response with isotropic shear behavior. The figure shows the traction on the vertical axis and the magnitudes of the normal and the shear separations along the two horizontal axes. The unshaded triangles in the two vertical coordinate planes represent the response under pure normal and pure shear deformation, respectively. All intermediate vertical planes (that contain the vertical axis) represent the damage response under mixed mode conditions with different mode mixes. The dependence of the damage evolution data on the mode mix can be defined either in tabular form or, in the case of an energy-based definition, analytically. The manner in which the damage evolution data are specified as a function of the mode mix is discussed later in this section.

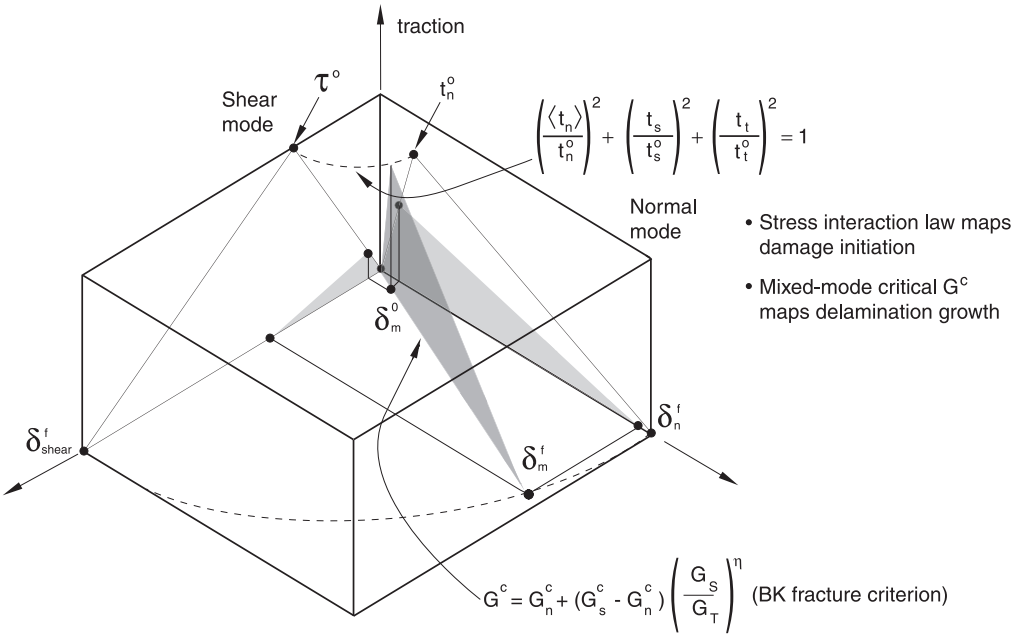


Figure 4: Illustration of mixed-mode response in cohesive elements.

Unloading subsequent to damage initiation is always assumed to occur linearly toward the origin of the traction-separation plane, as shown in *Figure 3*. Reloading subsequent to unloading also occurs along the same linear path until the softening envelope (line AB) is reached. Once the softening envelope is reached, further reloading follows this envelope as indicated by the arrow in *Figure 3*.

Evolution Based on Effective Displacement

You specify the quantity $\delta_m^f - \delta_m^o$ (that is, the effective displacement at complete failure, δ_m^f , relative to the effective displacement at damage initiation, δ_m^o, as shown in *Figure 3*) as a tabular function of the mode mix, temperature, and field variables. In addition, you also choose either a linear or an exponential softening law that defines the detailed evolution (between initiation and complete failure) of the damage variable, D, as a function of the effective displacement beyond damage initiation. Alternatively, instead of using linear or exponential softening, you can specify the damage variable, D, directly as a tabular function of the effective displacement after the initiation of damage, $\delta_m - \delta_m^o$; mode mix; temperature; and field variables.

Linear Damage Evolution

For linear softening (see *Figure 3*) Abaqus uses an evolution of the damage variable, D, that reduces (in the case of damage evolution under a constant mode mix, temperature, and field variables) to the expression proposed by Camanho and Davila (2002), namely:

$$D = \frac{\delta_m^f(\delta_m^{\max} - \delta_m^o)}{\delta_m^{\max}\left(\delta_m^f - \delta_m^o\right)}.$$

In the preceding expression and in all later references, $\delta_m^{\max}$ refers to the maximum value of the effective displacement attained during the loading history. The assumption of a constant mode mix at a material point between initiation of damage and final failure is customary for problems involving monotonic damage (or monotonic fracture).

Input File Usage: Use the following option to specify linear damage evolution:

*DAMAGE EVOLUTION, TYPE=DISPLACEMENT,
SOFTENING=LINEAR

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Quade Damage**, **Maxe Damage**, **Quads Damage**, or **Maxs Damage**: **Suboptions->Damage Evolution**: **Type:Displacement**: **Softening:Linear**

Exponential Damage Evolution

For exponential softening (see *Figure 5*) Abaqus uses an evolution of the damage variable, D, that reduces (in the case of damage evolution under a constant mode mix, temperature, and field variables) to

$$D = 1 - \left\{ \frac{\delta_m^o}{\delta_m^{\max}} \right\} \left\{ 1 - \frac{1 - \exp\left(-\alpha \left(\frac{\delta_m^{\max} - \delta_m^o}{\delta_m^f - \delta_m^o} \right) \right)}{1 - \exp(-\alpha)} \right\}.$$

In the expression above α is a nondimensional material parameter that defines the rate of damage evolution and $\exp(x)$ is the exponential function.

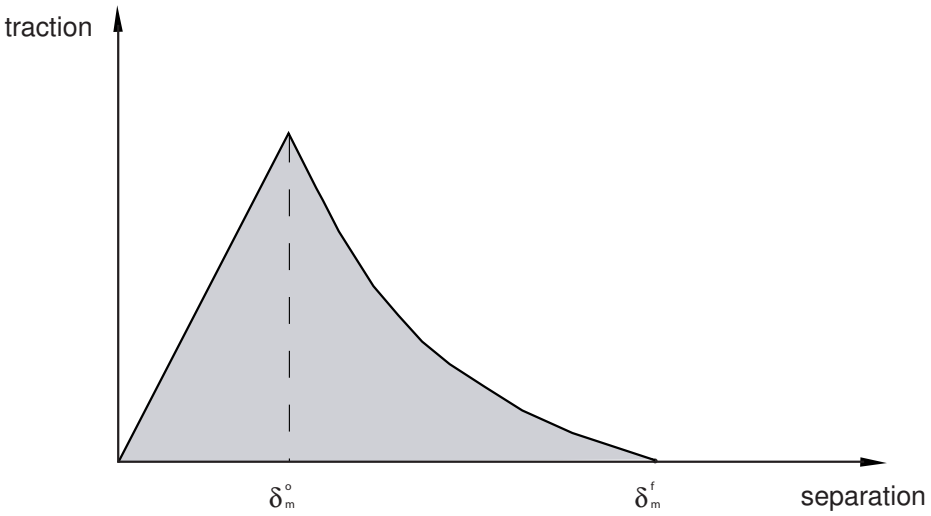


Figure 5: Exponential damage evolution.

Input File Usage: Use the following option to specify exponential softening:

*DAMAGE EVOLUTION, TYPE=DISPLACEMENT,
SOFTENING=EXPONENTIAL

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Quade Damage**, **Maxe Damage**, **Quads Damage**, or **Maxs Damage**: **Suboptions->Damage Evolution**: **Type:Displacement**: **Softening:Exponential**

Tabular Damage Evolution

For tabular softening you define the evolution of D directly in tabular form. D must be specified as a function of the effective displacement relative to the effective displacement at initiation, mode mix, temperature, and field variables.

Input File Usage: Use the following option to define the damage variable directly in tabular form:

**DAMAGE EVOLUTION*, TYPE=DISPLACEMENT,
SOFTENING=TABULAR

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Quade Damage**, **Maxe Damage**, **Quads Damage**, or **Maxs Damage**: **Suboptions->Damage Evolution**: **Type:Displacement**: **Softening:Tabular**

Evolution Based on Energy

Damage evolution can be defined based on the energy that is dissipated as a result of the damage process, also called the fracture energy. The fracture energy is equal to the area under the traction-separation curve (see *Figure 3*). You specify the fracture energy as a material property and choose either a linear or an exponential softening behavior. Abaqus ensures that the area under the linear or the exponential damaged response is equal to the fracture energy.

The dependence of the fracture energy on the mode mix can be specified either directly in tabular form or by using analytical forms as described below. When the analytical forms are used, the mode mix ratio is assumed to be defined in terms of energies.

Tabular Form

The simplest way to define the dependence of the fracture energy is to specify it directly as a function of the mode mix in tabular form.

Input File Usage: Use the following option to specify fracture energy as a function of the mode mix in tabular form:

**DAMAGE EVOLUTION*, TYPE=ENERGY,
MIXED MODE BEHAVIOR=TABULAR

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Quade Damage**, **Maxe Damage**, **Quads Damage**, or **Maxs Damage**: **Suboptions->Damage Evolution**: **Type:Energy**: **Mixed mode behavior:Tabular**

Power Law Form

The dependence of the fracture energy on the mode mix can be defined based on a power law fracture criterion. The power law criterion states that failure under mixed-mode conditions is governed by a power law interaction of the energies required to cause failure in the individual (normal and two shear) modes. It is given by

$$\left\{\frac{G_n}{G_n^C}\right\}^\alpha + \left\{\frac{G_s}{G_s^C}\right\}^\alpha + \left\{\frac{G_t}{G_t^C}\right\}^\alpha = 1.$$

The mixed-mode fracture energy $G^C = G_T$ when the above condition is satisfied. In other words,

$$G^C = 1 \Big/ \left(\left\{ \frac{m_1}{G_n^C} \right\}^\alpha + \left\{ \frac{m_2}{G_s^C} \right\}^\alpha + \left\{ \frac{m_3}{G_t^C} \right\}^\alpha \right)^{1/\alpha}.$$

You specify the quantities G_n^C, G_s^C, and G_t^C, which refer to the critical fracture energies required to cause failure in the normal, the first, and the second shear directions, respectively.

Input File Usage: Use the following option to define the fracture energy as a function of the mode mix using the analytical power law fracture criterion:

**DAMAGE EVOLUTION*, TYPE=ENERGY,
MIXED MODE BEHAVIOR=POWER LAW, POWER=α

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Quade Damage**, **Maxe Damage**, **Quads Damage**, or **Maxs Damage**: **Suboptions->Damage Evolution**: **Type:Energy**: **Mixed mode behavior:Power Law**: Toggle on **Power** and enter the exponent value

Benzeggagh-Kenane (BK) Form

The Benzeggagh-Kenane fracture criterion (Benzeggagh and Kenane, 1996) is particularly useful when the critical fracture energies during deformation purely along the first and the second shear directions are the same; that is, $G_s^C = G_t^C$. It is given by

$$G_n^C + \left(G_s^C - G_n^C\right) \left\{ \frac{G_S}{G_T} \right\}^\eta = G^C,$$

where $G_S = G_s + G_t$, $G_T = G_n + G_S$, and η is a material parameter. You specify G_n^C, G_s^C, and η.

Input File Usage: Use the following option to define the fracture energy as a function of the mode mix using the analytical BK fracture criterion:

**DAMAGE EVOLUTION*, TYPE=ENERGY,
MIXED MODE BEHAVIOR=BK, POWER=η

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Quade Damage**, **Maxe Damage**, **Quads Damage**, or **Maxs Damage**: **Suboptions->Damage Evolution**: **Type:Energy**: **Mixed mode behavior:Bk**: Toggle on **Power** and enter the exponent value

Linear Damage Evolution

For linear softening (see *Figure 3*) Abaqus uses an evolution of the damage variable, *D*, that reduces to

$$D = \frac{\delta_m^f \left(\delta_m^{\max} - \delta_m^o\right)}{\delta_m^{\max} \left(\delta_m^f - \delta_m^o\right)},$$

where $\delta_m^f = 2G^C / T_{\text{eff}}^o$ with T_{eff}^o as the effective traction at damage initiation. $\delta_m^{\max}$ refers to the maximum value of the effective displacement attained during the loading history.

Input File Usage: Use the following option to specify linear damage evolution:

*DAMAGE EVOLUTION, TYPE=ENERGY, SOFTENING=LINEAR

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Quade Damage**, **Maxe Damage**, **Quads Damage**, or **Maxs Damage**: **Suboptions->Damage Evolution**: **Type:Energy**: **Softening:Linear**

Exponential Damage Evolution

For exponential softening Abaqus uses an evolution of the damage variable, D, that reduces to

$$D = \int_{\delta_m^o}^{\delta_m^{\max}} \frac{T_{\text{eff}} d\delta}{G^C - G_o}.$$

In the expression above T_{eff} and δ are the effective traction and displacement, respectively. G_o is the elastic energy at damage initiation. In this case the traction might not drop immediately after damage initiation, which is different from what is seen in *Figure 5*.

Input File Usage: Use the following option to specify exponential softening:

*DAMAGE EVOLUTION, TYPE=ENERGY,
SOFTENING=EXPONENTIAL

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Damage for Traction-Separation Laws->Quade Damage**, **Maxe Damage**, **Quads Damage**, or **Maxs Damage**: **Suboptions->Damage Evolution**: **Type:Energy**: **Softening:Exponential**

Defining Damage Evolution Data as a Tabular Function of Mode Mix

As discussed earlier, the material data defining the evolution of damage can be tabular functions of the mode mix. The manner in which this dependence must be defined in Abaqus is outlined below for mode mix definitions based on energy and traction, respectively. In the following discussion it is assumed that the evolution is defined in terms of energy. Similar observations can also be made for evolution definitions based on effective displacement.

Mode Mix Based on Energy

For an energy-based definition of mode mix, in the most general case of a three-dimensional state of deformation with anisotropic shear behavior the fracture energy, G^C, must be defined as a function of $(m_2 + m_3)$ and $[m_3/(m_2 + m_3)]$. The quantity $(m_2 + m_3) = G_S/G_T$ is a measure of the fraction of the total deformation that is shear, while $[m_3/(m_2 + m_3)] = G_t/G_S$ is a measure of the fraction of the total shear deformation that is in the second shear direction. *Figure 6* shows a schematic of the fracture energy versus mode mix behavior.

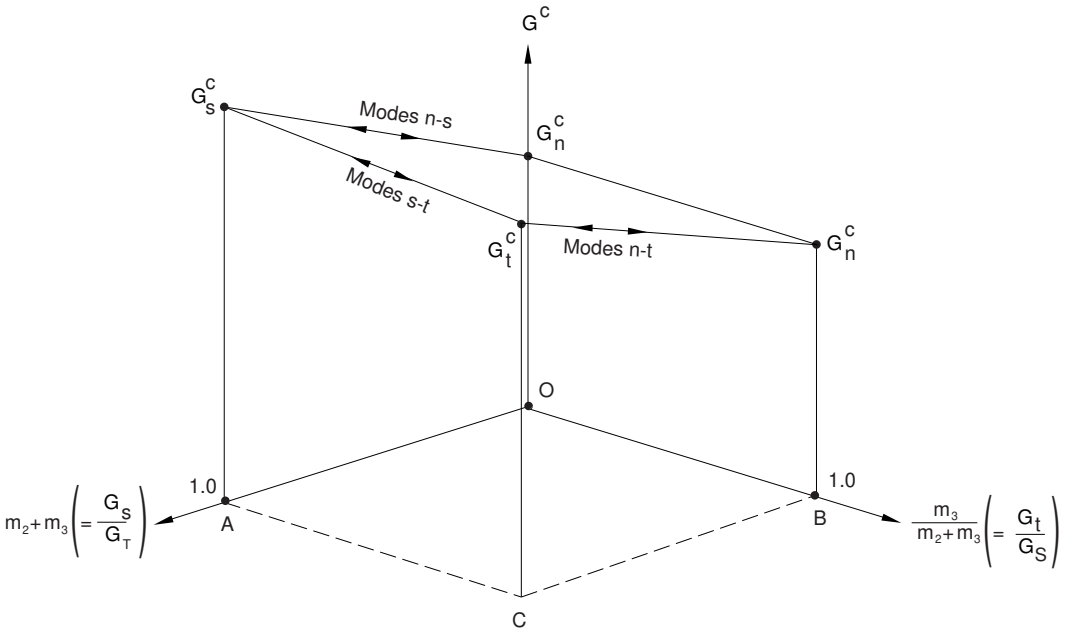


Figure 6: Fracture energy as a function of mode mix.

The limiting cases of pure normal and pure shear deformations in the first and second shear directions are denoted in *Figure 6* by G_n^C, G_s^C, and G_t^C, respectively. The lines labeled "Modes n-s," "Modes n-t," and "Modes s-t" show the transition in behavior between the pure normal and the pure shear in the first direction, pure normal and pure shear in the second direction, and pure shears in the first and second directions, respectively. In general, G^C must be specified as a function of $(m_2 + m_3)$ at various fixed values of $[m_3/(m_2 + m_3)]$. In the discussion that follows we refer to a data set of G^C versus $(m_2 + m_3)$ corresponding to a fixed $[m_3/(m_2 + m_3)]$ as a "data block." The following guidelines are useful in defining the fracture energy as a function of the mode mix:

- For a two-dimensional problem G^C needs to be defined as a function of m_2 ($m_3 = 0$ in this case) only. The data column corresponding to $[m_3/(m_2 + m_3)]$ must be left blank. Hence, essentially only one "data block" is needed.
- For a three-dimensional problem with isotropic shear response, the shear behavior is defined by the sum $(m_2 + m_3)$ and not by the individual values of m_2 and m_3. Therefore, in this case a single "data block" (the "data block" for $[m_3/(m_2 + m_3)] = 0$) also suffices to define the fracture energy as a function of the mode mix.
- In the most general case of three-dimensional problems with anisotropic shear behavior, several "data blocks" would be needed. As discussed earlier, each "data block" would contain G^C versus $(m_2 + m_3)$ at a fixed value of $[m_3/(m_2 + m_3)]$. In each "data block" $(m_2 + m_3)$ can vary between *0* and **1.0**. The case $(m_2 + m_3) = 0$ (the first data point in any "data block"), which corresponds to a purely normal mode, can never be achieved when $[m_3/(m_2 + m_3)] \neq 0$ (that is, the only valid point on line OB in *Figure 6* is the point O, which corresponds to a purely normal deformation). However, in the tabular definition of the fracture energy as a function of mode mix, this point simply serves to set a limit that ensures a continuous change in fracture energy as a purely normal state is approached from various combinations of normal and shear deformations. Hence, the fracture energy of the first data point in each "data block" must always be set equal to the fracture energy in a purely normal mode of deformation (G_n^C).

As an example of the anisotropic shear case, consider that you want to input three "data blocks" corresponding to fixed values of $[m_3/(m_2+m_3)] =$ 0., 0.2, and 1.0, respectively. For each of the three "data blocks," the first data point must be $(G_n^C, 0)$ for the reasons discussed above. The rest of the data points in each "data block" define the variation of the fracture energy with increasing proportions of shear deformation.

Mode Mix Based on Traction

The fracture energy needs to be specified in tabular form of G^C versus ϕ_1 and ϕ_2. Thus, G^C needs to be specified as a function of ϕ_1 at various fixed values of ϕ_2. A "data block" in this case corresponds to a set of data for G^C versus ϕ_1, at a fixed value of ϕ_2. In each "data block" ϕ_1 can vary from 0 (purely normal deformation) to 1 (purely shear deformation). An important restriction is that each data block must specify the same value of the fracture energy for $\phi_1 = 0$. This restriction ensures that the energy required for fracture as the traction vector approaches the normal direction does not depend on the orientation of the projection of the traction vector on the shear plane (see *Figure 2*).

Evaluating Damage When Multiple Criteria Are Active

When multiple damage initiation criteria and associated evolution definitions are used for the same material, each evolution definition results in its own damage variable, d_i, where the subscript *i* represents the *i*th damage system. The overall damage variable, *D*, is computed based on the individual d_i, as explained in *Evaluating Overall Damage When Multiple Criteria Are Active*, for damage in bulk materials.

Rate-Dependent Damage Evolution in Abaqus/Explicit

In Abaqus/Explicit you can define the damage evolution laws as a functions of the effective rate of separation. You can combine this definition with rate-dependent damage initiation to describe the rate-dependent delamination behavior for a cohesive interface.

Input File Usage: Use the following option to specify rate-dependent damage evolution:

**DAMAGE EVOLUTION*, RATE DEPENDENT

Abaqus/CAE Usage: Rate-dependent damage evolution is not supported in Abaqus/CAE.

Maximum Degradation and Choice of Element Removal

You have control over how Abaqus treats cohesive elements with severe damage. By default, the upper bound to the overall damage variable at a material point is $D_{max} = 1.0$. You can reduce this upper bound as discussed in *Controlling Element Deletion and Maximum Degradation for Materials with Damage Evolution*. You can control what happens to the cohesive element when the damage reaches this limit, as discussed below.

By default, once the overall damage variable reaches D_{max} at all of its material points and none of its material points are in compression, the cohesive elements, except for the pore pressure cohesive elements, are removed (deleted). See *Controlling Element Deletion and Maximum Degradation for Materials with Damage Evolution* for details. This element removal approach is often appropriate for modeling complete fracture of the bond and separation of components. Once removed, cohesive elements offer no resistance to subsequent penetration of the components, so it might be necessary to model contact between the components as discussed in *Defining Contact Between Surrounding Components*.

Alternatively, you can specify that a cohesive element should remain in the model even after the overall damage variable reaches D_{max}. In this case the stiffness of the element in tension and shear remains constant (degraded by a factor of $1 - D_{max}$ over the initial undamaged stiffness). This choice is appropriate if the cohesive elements must resist interpenetration of the surrounding components even after they have completely degraded in tension and shear (see *Defining Contact Between Surrounding Components*). In Abaqus/Explicit it is recommended that you suppress bulk viscosity in the cohesive elements by setting the scale factors for the linear and quadratic bulk viscosity parameters to zero using section controls (see *Section Controls*).

Uncoupled Transverse Shear Response

An optional linear elastic transverse shear behavior can be defined to provide additional stability to cohesive elements, particularly after damage has occurred. The transverse shear behavior is assumed to be independent of the regular material response and does not undergo any damage.

Input File Usage: Use the following options:

**COHESIVE SECTION*, RESPONSE=TRACTION SEPARATION
**TRANSVERSE SHEAR STIFFNESS*

Abaqus/CAE Usage: Transverse shear behavior is not supported in Abaqus/CAE for cohesive sections.

Viscous Regularization in Abaqus/Standard

Material models exhibiting softening behavior and stiffness degradation often lead to severe convergence difficulties in implicit analysis programs, such as Abaqus/Standard. A common technique to overcome some of these convergence difficulties is the use of viscous regularization of the constitutive equations, which causes the tangent stiffness matrix of the softening material to be positive for sufficiently small time increments.

The traction-separation laws can be regularized in Abaqus/Standard using viscosity by permitting stresses to be outside the limits set by the traction-separation law. The regularization process involves the use of a viscous stiffness degradation variable, D_v, which is defined by the evolution equation:

$$\dot{D}_v = \frac{1}{\mu}\left(D - D_v\right),$$

where μ is the viscosity parameter representing the relaxation time of the viscous system and D is the degradation variable evaluated in the inviscid backbone model. The damaged response of the viscous material is given as

$$\mathbf{t} = \left(1 - D_v\right)\bar{\mathbf{t}}.$$

Using viscous regularization with a small value of the viscosity parameter (small compared to the characteristic time increment) usually helps improve the rate of convergence of the model in the softening regime, without compromising results. The basic idea is that the solution of the viscous system relaxes to that of the inviscid case as $t/\mu \to \infty$, where t represents time. You can specify the value of the viscosity parameter as part of the section controls definition (see *Using Viscous Regularization with Cohesive Elements, Connector Elements, and Elements That Can Be Used with the Damage Evolution Models for Ductile Metals and Fiber-Reinforced Composites Including the Multiscale Model for Fiber-Reinforced Composites in Abaqus/Standard*). If the viscosity parameter is different from zero, output results of the stiffness degradation refer to the viscous value, D_v. The default value of the viscosity parameter is zero so that no viscous regularization is performed. Use of viscous regularization for improving the convergence behavior of delamination and debonding problems is discussed in *Delamination analysis of laminated composites* and *Analysis of skin-stiffener debonding under tension*.

The approximate amount of energy associated with viscous regularization over the whole model or over an element set is available using output variable ALLCD.

Output

In addition to the standard output identifiers available in Abaqus (*Abaqus/Standard Output Variable Identifiers* and *Abaqus/Explicit Output Variable Identifiers*), the following variables have special meaning for cohesive elements with traction-separation behavior:

NEEQ

Equivalent nominal strain, $\varepsilon_m = \frac{\delta_m}{T_0}$ (Abaqus/Explicit only). When the constitutive thickness of the cohesive layer is "artificially" set as $T_0 = 1.0$, the value of NEEQ is equal to the value of the effective displacement, δ_m.

NEEQR

Equivalent nominal strain rate, $\dot{\varepsilon}_m = \frac{\dot{\delta}_m}{T_0}$ (Abaqus/Explicit only). When the constitutive thickness of the cohesive layer is "artificially" set as $T_0 = 1.0$, the value of NEEQR is equal to the value of the effective rate of separation, $\dot{\delta}_m$. The effective rate of separation is used to evaluate rate-dependent damage initiation and evolution criteria.

STATUS

Status of element (the status of an element is 1.0 if the element is active, 0.0 if the element is not).

SDEG

Overall value of the scalar damage variable, *D*.

DMICRT

All damage initiation criteria components.

MAXSCRT

Maximum value of the nominal stress damage initiation criterion at a material point during the analysis. It is evaluated as $\max\left\{\frac{\langle t_n\rangle}{t_n^o}, \frac{t_s}{t_s^o}, \frac{t_t}{t_t^o}\right\}$.

MAXECRT

Maximum value of the nominal strain damage initiation criterion at a material point during the analysis. It is evaluated as $\max\left\{\frac{\langle \varepsilon_n\rangle}{\varepsilon_n^o}, \frac{\varepsilon_s}{\varepsilon_s^o}, \frac{\varepsilon_t}{\varepsilon_t^o}\right\}$.

MMIXDME

Mode mix ratio during damage evolution. If the energy-based mode mix is used, it is evaluated as $1 - m_1$; if the traction-based mode mix is used, it is evaluated as ϕ_1 . In general, it varies with time at a given integration point. This variable is set to -1.0 before initiation of damage.

MMIXDMI

Mode mix ratio at damage initiation. It is evaluated at the time of damage initiation at an integration point for the very first time as $1 - m_1$ if the energy-based mode mix is used; otherwise, if the traction-based mode mix is used, it is evaluated as ϕ_1. Following damage initiation, it remains constant with time at a given integration point. This variable is set to -1.0 before initiation of damage.

QUADSCRT

Maximum value of the quadratic nominal stress damage initiation criterion at a material point during the analysis. It is evaluated as $\left(\frac{\langle t_n\rangle}{t_n^o}\right)^2 + \left(\frac{t_s}{t_s^o}\right)^2 + \left(\frac{t_t}{t_t^o}\right)^2$.

QUADECRT Maximum value of the quadratic nominal strain damage initiation criterion at a material point during the analysis. It is evaluated as $\left(\frac{\langle \varepsilon_n \rangle}{\varepsilon_n^o}\right)^2 + \left(\frac{\varepsilon_s}{\varepsilon_s^o}\right)^2 + \left(\frac{\varepsilon_t}{\varepsilon_t^o}\right)^2$.

ALLCD The approximate amount of energy over the whole model or over an element set that is associated with viscous regularization in Abaqus/Standard. Corresponding output variables (such as CENER, ELCD, and ECDDEN) represent the energy associated with viscous regularization at the integration point level and element level (the last quantity represents the energy per unit volume in the element), respectively.

For the variables above that indicate whether a certain damage initiation criterion has been satisfied or not, a value that is less than 1.0 indicates that the criterion has not been satisfied, while a value of 1.0 or higher indicates that the criterion has been satisfied. If damage evolution is specified for this criterion, the maximum value of this variable does not exceed 1.0. However, if damage evolution is not specified for the initiation criterion, this variable can have values higher than 1.0. The extent to which the variable is higher than 1.0 can be considered to be a measure of the extent to which this criterion has been violated.

References

- Benzeggagh, M. L., and M. Kenane, “Measurement of Mixed-Mode Delamination Fracture Toughness of Unidirectional Glass/Epoxy Composites with Mixed-Mode Bending Apparatus,” *Composites Science and Technology*, vol. 56, pp. 439–449, 1996.
- Camanho, P. P., and C. G. Davila, “Mixed-Mode Decohesion Finite Elements for the Simulation of Delamination in Composite Materials,” *NASA/TM-2002–211737*, pp. 1–37, 2002.

Defining the Constitutive Response of Fluid within the Cohesive Element Gap

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Cohesive Elements*
- *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*
- **FLUID LEAKOFF*
- **GAP FLOW*
- *Adhesive joints and bonded interfaces*

Overview

The cohesive element fluid flow model:

- is typically used in geotechnical applications, where fluid flow continuity within the gap and through the interface must be maintained;
- enables fluid pressure on the cohesive element surface to contribute to its mechanical behavior, which enables the modeling of hydraulically driven fracture;
- enables modeling of an additional resistance layer on the surface of the cohesive element; and
- can be used only in conjunction with traction-separation behavior.

The features described in this section are used to model fluid flow within and across surfaces of pore pressure cohesive elements.

Defining Pore Fluid Flow Properties

The fluid constitutive response comprises:

- Tangential flow within the gap, which can be modeled with either a Newtonian or power law model; and
- Normal flow across the gap, which can reflect resistance due to caking or fouling effects.

The flow patterns of the pore fluid in the element are shown in *Figure 1*.

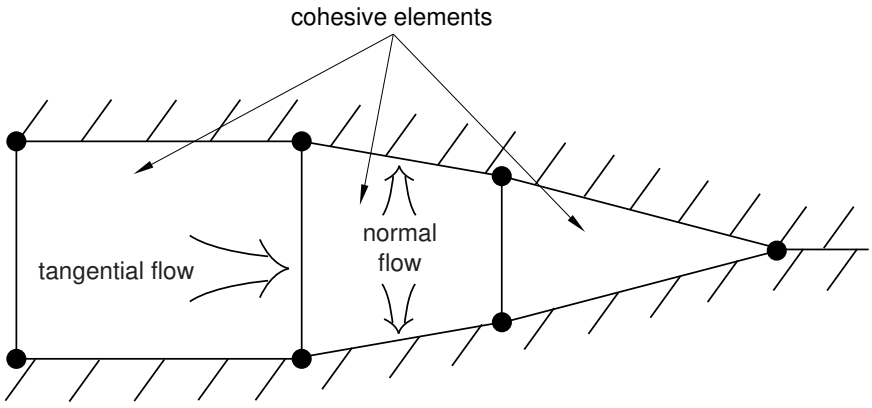


Figure 1: Flow within cohesive elements.

The fluid is assumed to be incompressible, and the formulation is based on a statement of flow continuity that considers tangential and normal flow and the rate of opening of the cohesive element.

Specifying the Fluid Flow Properties

You can assign tangential and normal flow properties separately.

Tangential Flow

By default, there is no tangential flow of pore fluid within the cohesive element. To allow tangential flow, define a gap flow property in conjunction with the pore fluid material definition.

Newtonian Fluid

In the case of a Newtonian fluid the volume flow rate density vector is given by the expression

$$\mathbf{q}d = -k_t \nabla p,$$

where k_t is the tangential permeability (the resistance to the fluid flow), ∇p is the pressure gradient along the cohesive element, and d is the gap opening.

In Abaqus the gap opening, d, is defined as

$$d = t_{curr} - t_{orig} + g_{init},$$

where t_{curr} and t_{orig} are the current and original cohesive element geometrical thicknesses, respectively; and g_{init} is the initial gap opening, which has a default value of 0.002.

Abaqus defines the tangential permeability, or the resistance to flow, according to Reynold's equation:

$$k_t = \frac{d^3}{12\mu},$$

where μ is the fluid viscosity and d is the gap opening. You can also specify an upper limit on the value of k_t.

Input File Usage: Use the following option to define the initial gap opening directly:

**SECTION CONTROLS*, INITIAL GAP OPENING

Use the following option to define the tangential flow in a Newtonian fluid:

**GAP FLOW*, TYPE=NEWTONIAN, KMAX

Abaqus/CAE Usage: Initial gap opening is not supported in Abaqus/CAE.

Property module: material editor: **Other**->**Pore Fluid**->**Gap Flow**: Type: **Newtonian**: Toggle on **Maximum Permeability** and enter the value of $k_{\max}$

Power Law Fluid

In the case of a power law fluid the constitutive relation is defined as

$$\tau = K\dot{\gamma}^{\alpha},$$

where τ is the shear stress, $\dot{\gamma}$ is the shear strain rate, K is the fluid consistency, and α is the power law coefficient. Abaqus defines the tangential volume flow rate density as

$$\mathbf{q}d = -\left(\frac{2\alpha}{1+2\alpha}\right)\left(\frac{1}{K}\right)^{\frac{1}{\alpha}}\left(\frac{d}{2}\right)^{\frac{1+2\alpha}{\alpha}}\|\nabla p\|^{\frac{1-\alpha}{\alpha}}\nabla p,$$

where d is the gap opening.

Input File Usage: **GAP FLOW*, TYPE=POWER LAW

Abaqus/CAE Usage: Property module: material editor: **Other->Pore Fluid->Gap Flow**: Type: **Power law**

Bingham Plastic Fluid

In the case of a Bingham plastic fluid the volume flow rate density vector is given by the expression

$$\mathbf{q}d = -\left(\frac{1}{3}\right)\left(\frac{1}{K}\right)\left(\frac{d}{2}\right)^3\nabla p\left(1-\frac{2\tau_0}{2d\|\nabla p\|}\right)^2\left(2+\frac{2\tau_0}{d\|\nabla p\|}\right),$$

where K is the fluid consistency, τ_0 is the yield stress, and d is the gap opening. The unyielded fluid is modeled as a Newtonian fluid with viscosity equal to $P_\tau \times K$, where P_τ has a default value of 10^7.

Input File Usage: **GAP FLOW*, TYPE=BINGHAM PLASTIC, PTAU=P_τ

Abaqus/CAE Usage: Property module: material editor: **Other->Pore Fluid->Gap Flow**: Type: **Bingham plastic**

Herschel-Bulkley Fluid

In the case of a Herschel-Bulkley fluid the volume flow rate density vector is given by the expression

$$\mathbf{q}d = -2\left[\left(\frac{\alpha}{1+\alpha}\right)\left(\frac{\alpha}{1+2\alpha}\right)\right]\left(\frac{1}{K}\right)^{\frac{1}{\alpha}}\left(\frac{d}{2}\right)^{\frac{1+2\alpha}{\alpha}}\|\nabla p\|^{\frac{1-\alpha}{\alpha}}\left(1-\frac{2\tau_0}{d\|\nabla p\|}\right)^{\frac{1+\alpha}{\alpha}}\left(\frac{1}{\alpha}+1+\frac{2\tau_0}{d\|\nabla p\|}\right)\nabla p,$$

where K is the fluid consistency, α is the power law coefficient, τ_0 is the yield stress, and d is the gap opening. The unyielded fluid is modeled as a Newtonian fluid with viscosity equal to $P_\tau \times K$, where P_τ has a default value of 10^7.

Input File Usage: **GAP FLOW*, TYPE=HERSCHEL-BULKLEY, PTAU=P_τ

Abaqus/CAE Usage: Property module: material editor: **Other->Pore Fluid->Gap Flow**: Type: **Herschel-Bulkley**

Normal Flow across Gap Surfaces

You can permit normal flow by defining a fluid leak-off coefficient for the pore fluid material. This coefficient defines a pressure-flow relationship between the cohesive element's middle nodes and their adjacent surface nodes. The fluid leak-off coefficients can be interpreted as the permeability of a finite layer of material on the cohesive element surfaces, as shown in *Figure 2*.

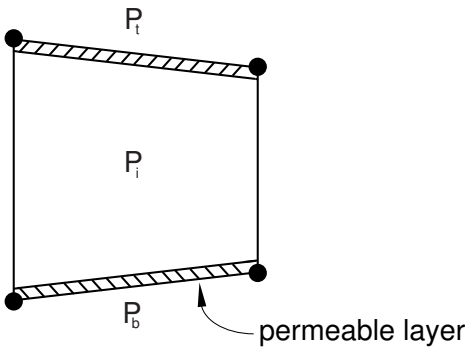


Figure 2: Leak-off coefficient interpretation as a permeable layer.

The normal flow is defined as

$$q_t = c_t \left(p_i - p_t\right)$$

and

$$q_b = c_b \left(p_i - p_b\right),$$

where q_t and q_b are the flow rates into the top and bottom surfaces, respectively; p_i is the midface pressure; and p_t and p_b are the pore pressures on the top and bottom surfaces, respectively.

Input File Usage: **FLUID LEAKOFF*

Abaqus/CAE Usage: Property module: material editor: **Other->Pore Fluid->Fluid Leakoff**: Type: **Coefficients**

Defining Leak-off Coefficients as a Function of Temperature and Field Variables

You can optionally define leak-off coefficients as functions of temperature and field variables.

Input File Usage: **FLUID LEAKOFF*, DEPENDENCIES

Abaqus/CAE Usage: Property module: material editor: **Other->Pore Fluid->Fluid Leakoff**: Type: **Coefficients**: Toggle on **Use temperature-dependent data** and select the number of field variables.

Defining Leak-off Coefficients in a User Subroutine

User subroutine *UFLUIDLEAKOFF* can also be used to define more complex leak-off behavior, including the ability to define a time accumulated resistance, or fouling, through the use of solution-dependent state variables.

Input File Usage: **FLUID LEAKOFF*, USER

Abaqus/CAE Usage: Property module: material editor: **Other->Pore Fluid->Fluid Leakoff**: Type: **User**

Tangential and Normal Flow Combinations

Table 1 shows the permitted combinations of tangential and normal flow and the effects of each combination.

Table 1: Effects of flow property definition combinations.

	Normal flow is defined	Normal flow is undefined
Tangential flow is defined	Tangential and normal flow are modeled.	Tangential flow is modeled. Pore pressure continuity is enforced between facing nodes in the cohesive element only when the element is closed. Otherwise, the surfaces are impermeable in the normal direction.
Tangential flow is undefined	Normal flow is modeled.	Tangential flow is not modeled. Pore pressure continuity is always enforced between facing nodes in the cohesive element.

Initially Open Elements

When the opening of the cohesive element is driven primarily by entry of fluid into the gap, you will have to define one or more elements as initially open, since tangential flow is possible only in an open element. Identify initially open elements as initial conditions.

Input File Usage: **INITIAL CONDITIONS*, TYPE=INITIAL GAP

Abaqus/CAE Usage: Initial gap definition is not supported in Abaqus/CAE.

Use of Unsymmetric Matrix Storage and Solution

The pore pressure cohesive element matrices are unsymmetric; therefore, unsymmetric matrix storage and solution may be needed to improve convergence (see *Matrix Storage and Solution Scheme in Abaqus/Standard*).

Additional Considerations

Your use of cohesive element fluid properties and your property values can impact your solution in some cases.

Large Coefficient Values

You must make sure that the tangential permeability or fluid leak-off coefficients are not excessively large. If either coefficient is many orders of magnitude higher than the permeability in the adjacent continuum elements, matrix conditioning problems may occur, leading to solver singularities and unreliable results.

Use in Total Pore Pressure Simulations

Definition of tangential flow properties may result in inaccurate results if the total pore pressure formulation is used and the hydrostatic pressure gradient contributes significantly to the tangential flow in the gap. The total pore pressure formulation is invoked if you apply gravity distributed loads to all elements in the model. The results will be accurate if the hydrostatic pressure gradient (i.e., the gravity vector) is perpendicular to the cohesive element.

Output

The following output variables are available when flow is enabled in pore pressure cohesive elements:

GFVR Gap fluid volume rate.

PFOPEN Fracture opening.

LEAKVRT Leak-off flow rate at element top.

ALEAKVRT Accumulated leak-off flow volume at element top.

LEAKVRB Leak-off flow rate at element bottom.

ALEAKVRB Accumulated leak-off flow volume at element bottom.

FLDVEL Material point fluid velocity.

Defining the Constitutive Response of Fluid Transitioning from Darcy Flow to Poiseuille Flow

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Cohesive Elements*
- *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*
- *Defining the Constitutive Response of Fluid within the Cohesive Element Gap*
- **FLUID LEAKOFF*
- **FLUID ID FIELD*
- **GAP FLOW*
- **PERMEABILITY*
- *Adhesive joints and bonded interfaces*
- *Element type assignment*

Overview

The cohesive element fluid flow model:

- is typically used in geotechnical applications, where fluid flow continuity within the cohesive element and through the interface must be maintained;
- supports the transition from Darcy flow to Poiseuille flow (gap flow) as damage in the element initiates and evolves;
- enables modeling of an additional resistance layer on the surface of the cohesive element to model fluid leakoff into the formation;
- enables fluid pressure on the cohesive element surface to contribute to its mechanical behavior, which enables the modeling of hydraulically driven fracture;
- can be used only in conjunction with traction-separation behavior;
- supports fluid flow continuity between intersecting layers of cohesive pore pressure elements;
- enables gravity-induced fluid flux modeling; and
- supports multiple fluid flow definition.

Defining Pore Fluid Flow Properties

The fluid constitutive response consists of the following:

- Tangential flow along the cohesive element midplane, which can be modeled as either Darcy or Poiseuille flow; and
- Normal flow (also referred to as leakoff) across the cohesive element, which can reflect resistance due to caking or fouling effects.

You assign the tangential and normal flow properties separately.

The flow patterns of the pore fluid in the element are shown in *Figure 1*. The fluid is assumed to be incompressible, and the formulation is based on a statement of flow continuity that considers tangential and normal flow and the rate of opening of the cohesive element.

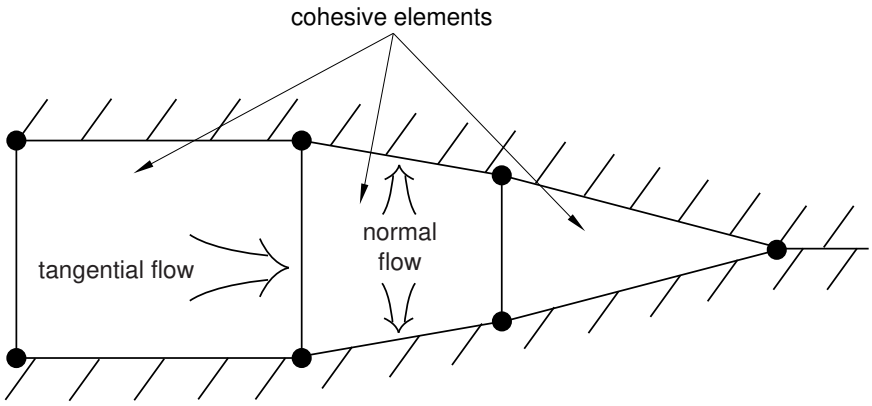


Figure 1: Flow within cohesive elements.

Tangential Flow

Tangential flow in a cohesive element will transition from Darcy flow to Poiseuille flow as damage in the element initiates and evolves. The transition is designed to approximate the changing nature of fluid flow through an initially undamaged porous material (Darcy flow) to flow in a crack (Poiseuille flow) as the material is damaged. You must specify the fluid constitutive response for both types of flow.

Gap Opening

The tangential flow equations for the cohesive element are solved within a gap along the length of the element. The gap opening, $\boldsymbol{d}$, is defined as

$$d = t_{curr} - t_{orig} + g_{init} = \hat{d} + g_{init},$$

where t_{curr} and t_{orig} are the current and original cohesive element geometrical thicknesses, respectively; g_{init} is the initial gap opening, which has a default value of 0.002; and $\hat{d}$ represents the physical crack opening that is used for Poiseuille flow once the element is damaged. A schematic illustration is shown in *Figure 2*. g_{init} is not a physical quantity. It is used by Abaqus/Standard to ensure that the flow equations can be solved robustly when the physical gap is closed (i.e., $\hat{d} = 0$). As $\hat{d}$ increases, the effect of g_{init} on the flow equations is diminished, as described in *Transition from Darcy Flow to Poiseuille Flow*.

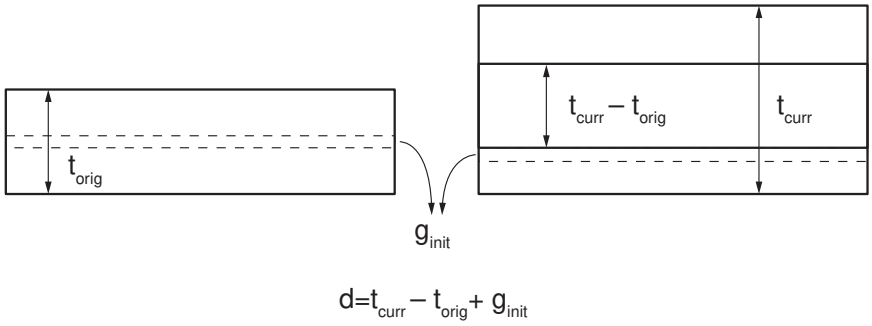


Figure 2: Cohesive elements gap opening.

Input File Usage: Use the following option to define the initial gap opening directly:

**SECTION CONTROLS*, INITIAL GAP OPENING

Abaqus/CAE Usage: Initial gap opening is not supported in Abaqus/CAE.

Darcy Flow

Darcy flow defines a simple relationship between the volumetric flow rate of a fluid and the fluid pressure gradient in a porous material. The relationship is defined by the expression

$$\mathbf{q}d = -\frac{kd}{\gamma_w}\nabla p,$$

where k is the permeability, ∇p is the pressure gradient along the cohesive element, d is the gap opening, and γ_w is the fluid specific weight.

Input File Usage: Use the following option to define fully saturated isotropic permeability:

**PERMEABILITY*, TYPE=ISOTROPIC, SPECIFIC=γ_w

Abaqus/CAE Usage: Property module: material editor: **Other**->**Pore Fluid**->**Permeability**: **Type: Isotropic**, **Specific weight of wetting liquid:** γ_w

Poiseuille Flow

In Abaqus/Standard Poiseuille flow within cohesive elements refers to the steady viscous flow between two parallel plates. For this flow, you can specify either a Newtonian fluid or a power law fluid.

Newtonian Fluid

In the case of a Newtonian fluid the volume flow rate density vector is given by the expression

$$\mathbf{q}d = -k_t \nabla p,$$

where k_t is the tangential permeability (the resistance to the fluid flow), ∇p is the pressure gradient along the cohesive element, and d is the gap opening.

Abaqus defines the tangential permeability, or the resistance to flow, according to Reynold's equation:

$$k_t = \frac{d^3}{12\mu},$$

where μ is the fluid viscosity and d is the gap opening. You can also specify an upper limit on the value of k_t.

Input File Usage: Use the following option to define the tangential flow in a Newtonian fluid:

**GAP FLOW*, TYPE=NEWTONIAN, KMAX

Abaqus/CAE Usage: Property module: material editor: **Other**->**Pore Fluid**->**Gap Flow**: Type: **Newtonian**: Toggle on **Maximum Permeability** and enter the value of $k_{\max}$

Power Law Fluid

In the case of a power law fluid the constitutive relation is defined as

$$\tau = K\dot{\gamma}^{\alpha},$$

where τ is the shear stress, $\dot{\gamma}$ is the shear strain rate, K is the fluid consistency, and α is the power law coefficient. Abaqus defines the tangential volume flow rate density as

$$\mathbf{q}d = -\left(\frac{2\alpha}{1+2\alpha}\right)\left(\frac{1}{K}\right)^{\frac{1}{\alpha}}\left(\frac{d}{2}\right)^{\frac{1+2\alpha}{\alpha}}\|\nabla p\|^{\frac{1-\alpha}{\alpha}}\nabla p,$$

where d is the gap opening.

Input File Usage: **GAP FLOW*, TYPE=POWER LAW

Abaqus/CAE Usage: Property module: material editor: **Other->Pore Fluid->Gap Flow**: Type: **Power law**

Bingham Plastic Fluid

In the case of a Bingham plastic fluid the volume flow rate density vector is given by the expression

$$\mathbf{q}d = -\left(\frac{1}{3}\right)\left(\frac{1}{K}\right)\left(\frac{d}{2}\right)^{3}\nabla p\left(1-\frac{2\tau_0}{2d\|\nabla p\|}\right)^{2}\left(2+\frac{2\tau_0}{d\|\nabla p\|}\right),$$

where K is the fluid consistency, τ_0 is the yield stress, and d is the gap opening. The unyielded fluid is modeled as a Newtonian fluid with viscosity equal to $P_\tau \times K$, where P_τ has a default value of 10^7.

Input File Usage: **GAP FLOW*, TYPE=BINGHAM PLASTIC, PTAU=P_τ

Abaqus/CAE Usage: The Bingham plastic fluid model is not supported in Abaqus/CAE.

Herschel-Bulkley Fluid

In the case of a Herschel-Bulkley fluid the volume flow rate density vector is given by the expression

$$\mathbf{q}d = -2\left[\left(\frac{\alpha}{1+\alpha}\right)\left(\frac{\alpha}{1+2\alpha}\right)\right]\left(\frac{1}{K}\right)^{\frac{1}{\alpha}}\left(\frac{d}{2}\right)^{\frac{1+2\alpha}{\alpha}}\|\nabla p\|^{\frac{1-\alpha}{\alpha}}\left(1-\frac{2\tau_0}{d\|\nabla p\|}\right)^{\frac{1+\alpha}{\alpha}}\left(\frac{1}{\alpha}+1+\frac{2\tau_0}{d\|\nabla p\|}\right)\nabla p,$$

where K is the fluid consistency, α is the power law coefficient, τ_0 is the yield stress, and d is the gap opening. The unyielded fluid is modeled as a Newtonian fluid with viscosity equal to $P_\tau \times K$, where P_τ has a default value of 10^7.

Input File Usage: **GAP FLOW*, TYPE=HERSCHEL-BULKLEY, PTAU=P_τ

Abaqus/CAE Usage: The Herschel-Bulkley fluid model is not supported in Abaqus/CAE.

Multiple Fluid Flow

Abaqus allows the use of multiple fluids in an analysis, up to a maximum of four fluid types. This capability is useful in situations where the spatial distribution of the different fluid types is known (precomputed) at all times during the analysis. In other words, the type of fluid at each node in the domain is known at all times during the analysis.

The definition of multiple fluids in an analysis involves assigning a unique integer value that acts as an identifier for each fluid type and associating each fluid identifier with a specific predefined field variable used to specify

the spatial distribution of the fluid as a function of time. You can utilize the fluid identifier to specify the viscosity of each fluid. Only the fluid viscosity can be different for different fluids. All other properties (for example, density, thermal, etc.) are assumed to be same for all the fluids. The field variable associated with a fluid type allows you to predefine the fluid type at a node as a function of time using amplitude definitions or as functions of both time and space utilizing user subroutine *UFIELD* or *USDFLD*.

You specify a value for each field variable, associated with a fluid identifier, at each node. It is recommended that you specify a value for the field variable (FV) such that $0 \leq FV \leq 1.0$. Abaqus assigns the fluid type at an integration point to be the one for which the field variable has the maximum value at that point. If all field variables have the same numerical value at an integration point, Abaqus assumes the first fluid (fluid identifier equal to 1) to be active.

Input File Usage: **GAP FLOW*, FLUID ID=*Fluid Id number*, DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC
**FIELD*, VARIABLE=*Field variable number*
**FLUID ID FIELD*
Fluid Id number, *Field variable number*

Abaqus/CAE Usage: Specifying multiple fluid flow is not supported in Abaqus/CAE.

Normal Flow across Gap Surfaces

You can permit normal flow by defining fluid leak-off coefficients for the pore fluid material. These coefficients define a pressure-flow relationship between the cohesive element's middle nodes and its adjacent surface nodes. The fluid leak-off coefficients can be interpreted as the permeability of a finite layer of material on the cohesive element surfaces, as shown in *Figure 3*.

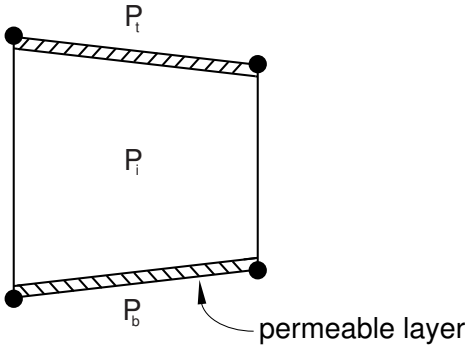


Figure 3: Leak-off coefficient interpretation as a permeable layer.

The normal flow is defined as

$$q_t = c_t \left(p_i - p_t\right)$$

and

$$q_b = c_b \left(p_i - p_b\right),$$

where q_t and q_b are the flow rates into the top and bottom surfaces, respectively; c_t and c_b are the fluid leak-off coefficients at the top and bottom element surfaces, respectively; p_i is the midface pressure; and p_t and p_b are the pore pressures on the top and bottom surfaces, respectively.

Input File Usage: **FLUID LEAKOFF*

Abaqus/CAE Usage: Property module: material editor: **Other**->**Pore Fluid**->**Fluid Leakoff**: Type: **Coefficients**

Defining Leak-off Coefficients as a Function of Temperature and Field Variables

You can optionally define leak-off coefficients as functions of temperature and field variables.

Input File Usage: **FLUID LEAKOFF*, DEPENDENCIES

Abaqus/CAE Usage: Property module: material editor: **Other**->**Pore Fluid**->**Fluid Leakoff**: Type: **Coefficients**: Toggle on **Use temperature-dependent data** and select the number of field variables.

Defining Leak-off Coefficients in a User Subroutine

User subroutine *UFLUIDLEAKOFF* can also be used to define more complex leak-off behavior, including the ability to define a time accumulated resistance, or fouling, through the use of solution-dependent state variables.

Input File Usage: **FLUID LEAKOFF*, USER

Abaqus/CAE Usage: Property module: material editor: **Other**->**Pore Fluid**->**Fluid Leakoff**: Type: **User**

Flow Flux Induced by Gravity

In the presence of a distributed gravity load the tangential flow rate density vector is given by the expression

$$\mathbf{q}d = -k_t\left(-\rho \mathbf{g}_t\right),$$

where k_t is the tangential permeability as defined above, $\mathbf{g}_t$ is the projection of the gravity vector onto the midsurface of the cohesive element, and ρ is the pore fluid density. For Darcy flow,

$$k_t = \frac{kd}{\gamma_w}.$$

For Poiseuille flow, in the case of a Newtonian fluid,

$$k_t = \frac{d^3}{12\mu}.$$

In the case of a power law fluid,

$$k_t = \left(\frac{2\alpha}{1+2\alpha}\right)\left(\frac{1}{K}\right)^{\frac{1}{\alpha}}\left(\frac{d}{2}\right)^{\frac{1+2\alpha}{\alpha}}\left\|\nabla p\right\|^{\frac{1-\alpha}{\alpha}}.$$

Input File Usage: Use the following option to specify the density of the pore fluid:

**DENSITY*, PORE FLUID

Abaqus/CAE Usage: Property module: material editor: **General**->**Density**

Transition from Darcy Flow to Poiseuille Flow

For a Newtonian fluid the transition from Darcy flow to Poiseuille flow as a function of the damage variable, D, is described by the expression

$$\mathbf{q}d = -\left(\left(1 - D\widehat{F}\left(\hat{d}\right)\right)\frac{k}{\gamma_w}g_{init} + D\widehat{F}\left(\hat{d}\right)\left(\frac{\left(\hat{d}\right)^3}{12\mu}\right)\right)\left(\nabla p - \rho\mathbf{g}_t\right),$$

$$\widehat{F}\left(\hat{d}\right) = \begin{cases} 0 & \hat{d} < 0 \\ \dfrac{\hat{d}}{g_{init}} & 0 \le \hat{d} \le g_{init} \\ 1 & g_{init} < \hat{d} \end{cases}.$$

The above relationship also supports a transition from Poiseuille flow back to Darcy flow as the physical gap, $\hat{d}$, in a damaged element closes. The flow transition equation for a power law fluid is obtained similarly.

Initially Open Elements

You can define an initial gap to identify elements that are fully damaged; that is, $D = 1.0$ at the integration points of the elements.

Input File Usage: Use the following option to define the initial gap directly:

**INITIAL CONDITIONS*, TYPE=INITIAL GAP
element number or element set, *omit values for D*

Abaqus/CAE Usage: Initial gap definition is not supported in Abaqus/CAE.

Assigning Initial Damage Values

You can define an initial gap to identify elements and assign D directly to the integration points. If you assign an initial damage variable to any of the integration points but not all of them, a value of $D = 0.0$ is assigned to the integration points to which you did not assign a value.

If an element set is used, you must ensure that all elements within the set have the proper uniform order of integration points.

Input File Usage: Use the following option to assign initial damage values:

**INITIAL CONDITIONS*, TYPE=INITIAL GAP
element number or element set, *D at each integration point*

Abaqus/CAE Usage: Initial gap definition is not supported in Abaqus/CAE.

Additional Considerations

Your use of cohesive element fluid properties and your property values can impact your solution in some cases.

Cohesive Offset

A cohesive offset (see *Cohesive Offset*)refers to a value of the separation in the normal direction, $\delta_n = \delta_n^{off}$, below which the resistance to additional closure $\left(\delta_n > 0, \dot{\delta}_n < 0\right)$ of the cohesive surfaces increases significantly. This capability is useful in preventing the closure of a fracture when changes in external loading conditions might otherwise result in closure. A practical application of this capability occurs during hydraulic fracturing with slurry, where solid proppant particles (the solid part of the slurry) help 'prop open' the fracture when external hydraulic pressure loads are removed.

Unsymmetric Matrix Storage and Solution

The pore pressure cohesive element matrices are unsymmetric; therefore, unsymmetric matrix storage and solution might be needed to improve convergence (see *Matrix Storage and Solution Scheme in Abaqus/Standard*).

Large Coefficient Values

You must make sure that the tangential permeability or fluid leak-off coefficients are not excessively large. If either coefficient is many orders of magnitude higher than the permeability in the adjacent continuum elements, matrix conditioning problems may occur, leading to solver singularities and unreliable results.

Meshing Requirement at Intersections of Cohesive Elements

When different layers of cohesive pore pressure elements intersect, a common midsurface node must be shared by all elements to support fluid flow continuity. *Figure 4* shows a two-dimensional mesh example of intersecting elements. Elements 10, 20, 30, and 40 share the same middle node, 100, at the intersecting point.

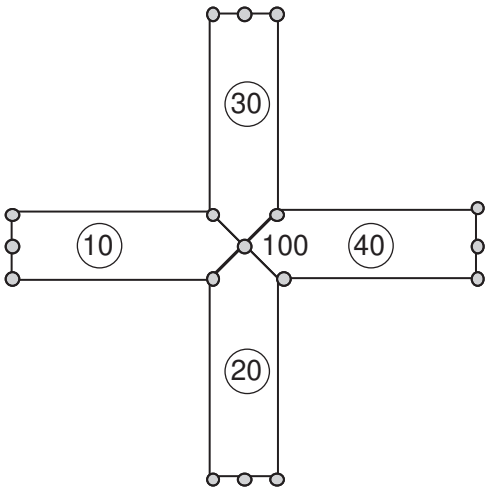


Figure 4: Meshing example for two-dimensional intersecting cohesive elements.

Pore Fluid Contact Property Consideration

To maintain pore pressure continuity between contact surfaces, you can define a pore fluid contact property (see *Pore Fluid Contact Properties*). If pore pressure degrees of freedom exist on both sides of a contact interface, the pore fluid contact property will enforce some pore pressure continuity. The default behavior is infinite permeability, which is equivalent to no resistance to the fluid flux between the two contact surfaces. If the top and bottom surfaces of the pore pressure cohesive elements are used to define the contact surface, the fluid flux should be controlled by the cohesive element definition. Therefore, the pore fluid contact property definition should be deactivated. You can remove its effects by setting the contact permeability to be zero and setting the cutoff gap fill to be zero.

Output

The following output variables are available when flow is enabled in pore pressure cohesive elements:

GFVR	Gap fluid volume rate.
PFOPEN	Fracture opening.
LEAKVRT	Leak-off flow rate at element top.
ALEAKVRT	Accumulated leak-off flow volume at element top.
LEAKVRB	Leak-off flow rate at element bottom.
ALEAKVRB	Accumulated leak-off flow volume at element bottom.
FLDVEL	Material point fluid velocity.
FLVF	When multiple fluids are defined, the components of FLVF represent the volume fraction of each fluid type. In particular, FLVF_*i* refers to the volume fraction of fluid type *i*.

Defining Thermal Convection for a Cohesive Element

Products: Abaqus/Standard Abaqus/CAE

References:

- **DENSITY*
- **SPECIFIC HEAT*
- **CONDUCTIVITY*
- **GAP CONVECTION*
- **MATERIAL*

Overview

You can define convective heat transfer with gap flow as well as thermal convection between the gap fluid and both the top and bottom surfaces of a coupled temperature-pore pressure cohesive element.

Thermal convection modeling using coupled temperature-pore pressure cohesive elements:

- can be performed in coupled pore fluid diffusion/stress analyses (*Coupled Pore Fluid Diffusion and Stress Analysis*);
- can also involve convective heat transfer between the gap fluid and both the top and bottom surfaces of a cohesive element; and
- can be specified by a material model.

Modeling Thermal Convection Together with Gap Fluid Flow within a Cohesive Element

The thermal convection contributes toward the energy balance. The energy equation is given by:

$$\rho_f c_f (d\theta)_{,t} + \rho_f c_f \nabla\cdot(d\mathbf{q}\theta) - \nabla\cdot(dk\nabla\theta) + \rho_f c_f [c_t(p_i - p_t)\theta_t + c_b(p_i - p_b)\theta_b] + h_t(\theta_i - \theta_t) + h_b(\theta_i - \theta_b) = 0,$$

where

ρ_f
: is the fluid density,

c_f
: is the specific heat of the fluid,

k
: is the conductivity of the fluid,

h_t, h_b
: are the heat convection coefficients for the top and the bottom surfaces respectively; they are assumed to have the same value (that is, $h_t = h_b = h$),

θ
: is the temperature of the gap flow,

θ_t
: is the temperature on the top surface, and

θ_b
is the temperature on the bottom surface.

All of the other variables (such as $\mathbf{q}$, d, p_i, p_t, p_b, c_t, and c_b) are defined in *Defining the Constitutive Response of Fluid Transitioning from Darcy Flow to Poiseuille Flow*.

Numerical solutions to the thermal convection-diffusion equation become increasingly difficult and prone to oscillations as convection begins to dominate diffusion. To alleviate these numerical issues, the Petrov-Galerkin scheme is used by default to formulate the equations.

Specifying Fluid Density

You can define the gap fluid density.

Input File Usage: **DENSITY*, PORE FLUID

Abaqus/CAE Usage: Specifying the gap fluid density is not supported in Abaqus/CAE.

Specifying Specific Heat of the Fluid

You can define the specific heat of the pore fluid in a porous medium.

Input File Usage: **SPECIFIC HEAT*, PORE FLUID

Abaqus/CAE Usage: Specifying the specific heat of the pore fluid in a porous medium is not supported in Abaqus/CAE.

Specifying Conductivity of the Fluid

You can define the conductivity of the pore fluid in a porous medium.

Input File Usage: **CONDUCTIVITY*, PORE FLUID

Abaqus/CAE Usage: Specifying the conductivity of the pore fluid in a porous medium is not supported in Abaqus/CAE.

Specifying Heat Convection between the Fluid and Both the Top and Bottom Surfaces

The heat convection coefficient is defined as

$$h = \frac{\mathbf{Nu}}{2}\frac{k}{d},$$

where $\mathbf{Nu}$ is the Nusselt number. You can specify the Nusselt number as a function of temperature and field variables.

Input File Usage: Use the following option to define a constant flux solution for convective heat transfer ($\mathbf{Nu} = 8.235$):

**GAP CONVECTION*, TYPE=FLUX

Use the following option to define a constant temperature solution for convective heat transfer ($\mathbf{Nu = 7.54}$):

**GAP CONVECTION*, TYPE=TEMPERATURE

Use the following option to specify the Nusselt number as a function of temperature and/or field variables:

**GAP CONVECTION*, TYPE=TABULAR

Abaqus/CAE Usage: Property module: material editor: **Other->Pore Fluid->Gap Convection**: **Type**: **FLUX**, **TEMPERATURE**, or **TABULAR**

Defining a Thermal Interaction for a Cohesive Element

Products: Abaqus/Standard Abaqus/CAE

References:

- *GAPCON*
- **GAP CONDUCTANCE*
- **GAP RADIATION*

Overview

You can define thermal interaction between the top and bottom surfaces of a coupled temperature-displacement cohesive element.

A thermal interaction between the top and bottom surfaces of a coupled temperature-displacement cohesive element:

- can be included in a thermal-stress analysis (*Fully Coupled Thermal-Stress Analysis*);
- can involve conductive heat transfer between the top and bottom surfaces;
- can involve radiative heat transfer between the top and bottom surfaces when the surfaces are separated by a narrow gap; and
- can be specified by a material model.

Specifying Thermal Conductance of a Cohesive Element

The thermal conductance between the top and bottom surfaces of a coupled temperature-displacement cohesive element can be defined by specifying gap conductance as part of the material definition.

Input File Usage: Use the following options to define the thermal conductance with a material model:

**COHESIVE SECTION*, ELSET=*name*, MATERIAL=*name*
**MATERIAL*, NAME=*name*
**GAP CONDUCTANCE*

Abaqus/CAE Usage: Use the following option to define the thermal conductance with a material model:

Property module: material editor: **Thermal->Gap Conductance**

Modeling Conductance between the Top and Bottom Surfaces of a Cohesive Element

The conductive heat transfer between the top and bottom surfaces of a cohesive element is assumed to be defined by

$$q = k(\theta_A - \theta_B),$$

where q is the heat flux per unit area crossing the cohesive element from point A on the top surface to point B on the bottom surface, θ_A and θ_B are the temperatures of the points on the surfaces, and k is the gap conductance. Point A is a node on the top surface, and point B is the corresponding node on the bottom surface.

You can define k directly or in user subroutine *GAPCON*.

Defining Gap Conductance Directly

When defining k directly, define it as

$$k = k\left(d, \bar{\theta}, \overline{|\dot{m}|}, \bar{f}_\gamma\right),$$

where

d

is the normal separation between A and B,

$\bar{\theta} = \frac{1}{2}\left(\theta_A + \theta_B\right)$

is the average of the surface temperatures at A and B,

$\overline{|\dot{m}|}$

is not considered here and should be set as zero, and

$\bar{f}_\gamma = \frac{1}{2}\left(f_\gamma^A + f_\gamma^B\right)$

is the average of any predefined field variables at A and B.

Defining Gap Conductance as a Function of Normal Separation

You can create a table of data defining the dependence of k on the variables listed above. The default in Abaqus is to make k a function of the separation d. When k is a function of separation, d, the tabular data must start at zero separation and define k as d increases. At least two pairs of k-d points must be given to define k as a function of the separation. The value of k drops to zero immediately after the last data point, so there is no heat conductance when the separation is greater than the value corresponding to the last data point.

Input File Usage: **GAP CONDUCTANCE*
k, d, $\bar{\theta}$

Abaqus/CAE Usage: Property module: material editor: **Thermal->Gap Conductance**: enter table data

Defining Gap Conductance as a Function of Predefined Field Variables

In addition to the dependencies mentioned previously, the gap conductance can be dependent on any number of predefined field variables, $\bar{f}_\gamma$. To make the gap conductance depend on field variables, at least two data points are required for each field variable value.

Input File Usage: **GAP CONDUCTANCE*, DEPENDENCIES=*n*
k, d, $\bar{\theta}$, $\overline{|\dot{m}|}$, $\bar{f}_\gamma$

Abaqus/CAE Usage: Property module: material editor: **Thermal->Gap Conductance**: **Number of field variables**=*n*

Defining the Gap Conductance in User Subroutine GAPCON

You can define k in user subroutine *GAPCON*. In this case there is greater flexibility in specifying the dependencies of k. It is no longer necessary to define k as a function of the average of the two surface's temperatures, mass flow rates, or field variables:

$$k = k\left(d, p, \theta_A, \theta_B, |\dot{m}|_A, |\dot{m}|_B, f_\gamma^A, f_\gamma^B\right).$$

The pressure and mass flow rates in user subroutine *GAPCON* are not used to model conductance in cohesive elements, and the variables should be set to zero.

Input File Usage: **GAP CONDUCTANCE*, USER

Abaqus/CAE Usage: Property module: material editor: **Thermal->Gap Conductance**: User-defined

Modeling Radiation between Surfaces When the Gap Is Small

Abaqus assumes that radiative heat transfer between closely spaced surfaces occurs in the direction of the normal between the top and bottom surfaces.

The gap radiation functionality in Abaqus is intended for modeling radiation between surfaces across a narrow gap. A more general capability for modeling radiation is available in Abaqus/Standard (see *Cavity Radiation in Abaqus/Standard*).

Radiative heat transfer is defined as a function of normal separation between the top and bottom surfaces through the effective view factor. Abaqus maintains the radiative heat flux even when the surfaces are in contact. This causes only a minor inaccuracy since normally the heat flux from conduction is much larger than the radiative heat flux.

Abaqus defines the heat flow per unit surface area between corresponding points as

$$q = C\left[\left(\theta_A - \theta^Z\right)^4 - \left(\theta_B - \theta^Z\right)^4\right],$$

where q is the heat flux per unit surface area crossing the gap at this point from surface A to surface B, θ_A and θ_B are the temperatures of the two surfaces, θ^Z is the absolute zero on the temperature scale being used, and the coefficient C is given by

$$C = \frac{F\sigma}{1/\epsilon_A + 1/\epsilon_B - 1},$$

where σ is the Stefan-Boltzmann constant, ϵ_A and ϵ_B are the surface emissivities, and F is the effective view factor, which corresponds to viewing the main surface from the secondary surface.

The view factor F must be defined as a function of the separation, d, and should have a value between 0.0 and 1.0. At least two pairs of F-d points are required to define the view factor, and the tabular data must start at zero separation (closed gap) and define the view factor as the separation increases. The value of F drops to zero immediately after the last data point, so there is no radiative heat transfer when the separation is greater than the value corresponding to the last data point.

Input File Usage: **GAP RADIATION*
ϵ_A, ϵ_B
$F_0, 0.$
F_1, d_1
…

Abaqus/CAE Usage: Property module: material editor: **Thermal->Gap Radiation**: enter data

Specifying the Value of Absolute Zero

You must specify the value of θ^Z.

Input File Usage: **PHYSICAL CONSTANTS*, ABSOLUTE ZERO=θ^Z

Abaqus/CAE Usage: Any module: **Model->Edit Attributes->*model_name***: **Absolute zero temperature=θ^Z**

Specifying the Stefan-Boltzmann Constant

You must specify the Stefan-Boltzmann constant, σ.

Input File Usage: **PHYSICAL CONSTANTS*, STEFAN BOLTZMANN=σ

Abaqus/CAE Usage: Any module: **Model->Edit Attributes->*model_name***: **Stefan-Boltzmann constant=σ**

Improving Convergence in Abaqus/Standard

Because the heat flux due to radiation is a strongly nonlinear function of the temperature, the radiation equations are strongly nonsymmetric and using the unsymmetric matrix storage and solution scheme for the step may improve the convergence rate in Abaqus/Standard (see *Defining an Analysis*).

Defining Slurry Transport and Placement within the Cohesive Element Gap

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Cohesive Elements*
- *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*
- *Defining the Constitutive Response of Fluid Transitioning from Darcy Flow to Poiseuille Flow*
- **FLUID LEAKOFF*
- **GAP FLOW*
- **PERMEABILITY*
- *Adhesive joints and bonded interfaces*
- *Element type assignment*

Overview

Slurry transport and placement within the cohesive element gap:

- is typically used in geotechnical applications where the continuity of the slurry mixture and its components within the cohesive element and through the interface must be maintained;
- supports the transition from Darcy flow to Poiseuille flow (gap flow) as damage in the element initiates and evolves;
- enables modeling of an additional resistance layer on the surface of the cohesive element to model fluid leakoff into the formation, with or without considering the evolution of filter cake;
- enables fluid pressure on the cohesive element surface to contribute to its mechanical behavior, which enables the modeling of hydraulically driven fracture;
- can be used only in conjunction with traction-separation cohesive response;
- supports slurry flow continuity between intersecting layers of cohesive pore pressure elements;
- enables proppant settling due to gravity and wall drag effects;
- enables the modeling of hydraulic fracture propped by solid particles; and
- allows multiple fluid definition.

Modeling Slurry Flow

The slurry transport and placement capabilities in Abaqus/Standard model transport of solids (proppants)-laden Newtonian and non-Newtonian slurries within the gap between the two faces of a cohesive element. The slurry advects into the gap region, as it opens up due to hydraulic pressure. Subsequently, the particles in the slurry settle down within the gap through a process that is mediated by wall drag and gravity effects. The particles help prop open the gap if it tries to close down due to changes in external conditions.

This complex set of phenomena within the cracked surfaces can be described in terms of the mean tangential flow of the slurry mixture, the advection of the proppant particles with the carrying fluid, and the settling of the proppant particles within the gap due to wall drag and gravity effects. The process is governed by the continuity of the slurry mixture (a mixture of the proppant particles and the carrying fluid) that describes the mean flow of the slurry, the continuity of the carrier fluid, a description of the settling phenomenon in terms of the relative velocity of the particles with respect to the carrying fluid, and leakoff (normal flow) of the carrying fluid into the surrounding formation.

Tangential Slurry Flow

The continuity equation for the mean flow of the slurry mixture within the gap can be written as

$$\frac{\partial d}{\partial t} + \nabla \cdot (d\boldsymbol{V}_m) + q_l = 0,$$

where $\boldsymbol{d}$ is the gap opening, $\boldsymbol{V_m}$ is the *mean velocity* of the slurry, and $\boldsymbol{q_l}$ is the leak-off flow rate. The continuity equation for the carrier fluid can be written as

$$\frac{\partial[(1-C_v)d]}{\partial t} + \nabla \cdot [(1 - C_v)d\boldsymbol{V}_w] + q_l = 0,$$

where $\boldsymbol{C_v}$ is the concentration of proppant particles in the slurry, defined as the volume ratio of the solid phase to that of the slurry phase (subsequently referred to as the slurry concentration), and $\boldsymbol{V_w}$ is the velocity vector of the carrier fluid. In formulating the above equation, it is assumed that the carrier fluid and solid particles move with different velocity vectors, $\boldsymbol{V_w}$ and $\boldsymbol{V_s}$, respectively, and that they are related to the mean slurry velocity and a settling (or slip) velocity, $\boldsymbol{V_{s/w}}$, as

$$\begin{cases} \boldsymbol{V_w} = \boldsymbol{V_m} + C_v \boldsymbol{V_{s/w}} \\ \boldsymbol{V_s} = \boldsymbol{V_m} - (1 - C_v)\, \boldsymbol{V_{s/w}} \end{cases},$$

where the settling velocity $\boldsymbol{V_{s/w}} = \boldsymbol{V_w} - \boldsymbol{V_s}$ represents the relative velocity of the proppant particles with respect to the carrier fluid. Using the above velocity split, the continuity equation for the carrier fluid can be rewritten as

$$\frac{\partial[(1-C_v)d]}{\partial t} + \nabla \cdot [(1 - C_v)d\boldsymbol{V}_m] + q_l + \nabla \cdot \left[C_v(1 - C_v)d\boldsymbol{V}_{s/w}\right] = 0.$$

The above system of equations is augmented by a relation of the following form, which relates the mean slurry flux, $\mathbf{q}$, and the gradient of the fluid pressure field (assuming non-Newtonian constitutive behavior of the slurry):

$$\mathbf{q}d = f\left(K, \alpha, \tau_0, d, \nabla p\right),$$

where $\boldsymbol{K}$ is the fluid consistency, $\boldsymbol{\alpha}$ is the power law coefficient, $\boldsymbol{\tau_0}$ is the yield stress, and $\boldsymbol{d}$ is the gap opening. The above system of equations helps determine the fluid pressure field that drives the slurry flow through the gap, the mean slurry flow velocity field, and the slurry concentration, $\boldsymbol{C_v}$ (nodal degree of freedom 31), accounting for both leakoff and settling.

Abaqus assumes that the homogenized slurry flow properties can be defined in a manner that is similar to the corresponding definitions for the flow of a pure fluid with the cohesive gap (*Defining the Constitutive Response of Fluid Transitioning from Darcy Flow to Poiseuille Flow*), except that the flow properties can be functions of the slurry concentration.

Input File Usage: Use the following option to define the tangential flow of a slurry mixture with a specified fluid type:

**GAP FLOW*, TYPE=*type_name*, SLURRY

Abaqus/CAE Usage: Defining the tangential flow of the slurry mixture is not supported in Abaqus/CAE.

Proppant Settling

By default, Abaqus assumes that the particle settling phenomena is mainly due to gravity (solid particles and carrier fluid usually have different densities) and wall drag effects (hindrance due to the narrowness of the gap). However you can suppress the settling completely (thereby modeling proppant transport due to advection only), or you can define the settling velocity field directly through user subroutine *USETTLING*.

Input File Usage: Use one of the following options to account for settling effects using the Abaqus built-in model:

**GAP FLOW*, TYPE=*type_name*, SLURRY

**GAP FLOW*, TYPE=*type_name*, SLURRY, SETTLING

Use the following option to suppress settling effects:

**GAP FLOW*, TYPE=*type_name*, SLURRY, SETTLING=NONE

Use the following option to define the settling velocity field directly through user subroutine *USETTLING*:

**GAP FLOW*, TYPE=*type_name*, SLURRY, SETTLING=USER

Abaqus/CAE Usage: Specifying proppant particle settling is not supported in Abaqus/CAE.

Gap Particle Size Ratio

By default, Abaqus assumes that the ratio of the opening between the gap surfaces to the particle size must be greater than 1.0 for the slurry particles to enter the gap. Slurry calculations inside the gap are not activated by default if the calculated ratio between these quantities is less than 1.0. Alternatively, you can specify a nondefault minimum ratio of the gap opening to the particle size.

Input File Usage: Use the following option to define the density of carrier fluid, the density of the proppant particles, and the diameter of proppant particles (it is assumed that all proppant particles in the slurry have the same diameter), respectively:

**DENSITY*, PORE FLUID, SLURRY

Use the following option to define the ratio of the minimum gap opening to the particle size directly:

**SECTION CONTROLS*, MIN GAP PARTICLE RATIO

Abaqus/CAE Usage: Defining the density of the carrier fluid and the proppant particles and the gap-particle diameter ratio of the slurry is not supported in Abaqus/CAE.

Multiple Fluid Flow

Abaqus allows the use of multiple fluids in an analysis, up to a maximum of four fluid types. This capability is useful in situations where the spatial distribution of the different fluid types is known (precomputed) at all times during the analysis. In other words, the type of fluid at each node in the domain is known at all times during the analysis.

The definition of multiple fluids in an analysis involves assigning a unique integer value that acts as an identifier for each fluid type and associating each fluid identifier with a specific predefined field variable used to specify the spatial distribution of the fluid as a function of time. You can use the fluid identifier to specify the viscosity of each fluid. Only the fluid viscosity can be different for different fluids. All other properties (for example, density, thermal, etc.) are assumed to be same for all the fluids. The field variable associated with a fluid type allows you to predefine the fluid type at a node as a function of time using amplitude definitions or as functions of both time and space utilizing user subroutine *UFIELD* or *USDFLD*.

You specify a value for each field variable, associated with a fluid identifier, at each node. It is recommended that you specify a value for the field variable ($\boldsymbol{FV}$) such that $\mathbf{0} \leq \boldsymbol{FV} \leq \mathbf{1.0}$. Abaqus assigns the fluid type at an integration point to be the one for which the field variable has the maximum value at that point. If all field

variables have the same numerical value at an integration point, the first fluid (fluid identifier 1) is assumed to be active.

Input File Usage: **GAP FLOW*, FLUID ID=*Fluid Id number*, DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC
**FIELD*, VARIABLE=*Field variable number*
**FLUID ID FIELD*
Fluid Id number, *Field variable number*

Abaqus/CAE Usage: Specifying multiple fluid flow is not supported in Abaqus/CAE.

Stability of Tangential Flow

A Petrov-Galerkin scheme (see *Convection/diffusion*) is used to discretize the governing equation for the slurry concentration because the slurry transport is a highly advection dominant phenomenon. Although diffusion effects are assumed to be negligible, you can define a small amount of diffusivity to provide numerical stabilization.

You must be careful to select a value of the diffusivity coefficient such that the fundamental nature of the transport problem is not modified from convection dominated to diffusion dominated. A good measure of the relative dominance of convective versus diffusive contributions is the local Péclet number, γ, which is defined as:

$$\gamma = |\mathbf{v}| \frac{\Delta l}{D},$$

where $|\mathbf{v}|$ is the magnitude of the velocity vector, Δl is a characteristic element length in the direction of flow, and D is the diffusivity of the proppant. Large values of γ indicate that convection dominates over diffusion on the spatial scale defined by the element size Δl.

Input File Usage: Use the following option to define the diffusivity of proppant particles in a slurry flow:

**DIFFUSIVITY*, SLURRY

Abaqus/CAE Usage: Defining the diffusivity of proppant particles of a slurry flow is not supported in Abaqus/CAE.

Normal Flow across Gap Surfaces

The normal flow across the gap surfaces can be modeled using one of the following approaches:

- A simple leak-off behavior that requires leak-off coefficients on the top and bottom surfaces to define the flux of the normal flow.
- A physics-based filtration model that accounts for the formation and evolution of filter cake layers on the two cohesive surfaces.

The physics-based approach has the following flavors (*Outmans, 1963*), which are applicable only when the slurry concentration degree of freedom is active in the analysis:

- A dynamic filtration model that assumes the filter cake to be compressible.
- A static filtration model that assumes the filter cake to be incompressible.

Therefore, you can model the normal flow across the gap surfaces using one of the following approaches:

- the simple leak-off model in which you define the leak-off coefficients directly;

- a physics-based model that accounts for the evolution of the filter cake thickness, filter cake viscosity, and filter case permeability; or
- - through user subroutine *UFLUIDLEAKOFF*, which allows you to define more complex leak-off behavior (including the ability to define a time-accumulated resistance, or fouling, through the use of solution-dependent state variables).

Simple Leak-Off Behavior

The simple leak-off behavior assumes that the flux of the normal flow across the top and the bottom surfaces is defined by a simple pressure-flow relationship (for more information, see *Normal Flow across Gap Surfaces*).

The simple leak-off model does not account for the dependence of the leak-off behavior on the slurry concentration but can still provide a reasonable description of the normal flow across gaps in some situations.

Input File Usage: Use the following option to define the fluid leak-off coefficients at the top and the bottom surfaces of the cohesive element directly, (optionally) as tabular functions of temperature:

**FLUID LEAKOFF*

Use the following option to define the fluid leak-off coefficients at the top and the bottom surfaces of the cohesive element directly, (optionally) as tabular functions of (optionally) temperature and field variables:

**FLUID LEAKOFF*, DEPENDENCIES

Use the following option to define the fluid leak-off coefficients using user subroutine *UFLUIDLEAKOFF*:

**FLUID LEAKOFF*, USER

Abaqus/CAE Usage: Property module: material editor: **Other->Pore Fluid->Fluid Leakoff**: Type: **Coefficients**

Property module: material editor: **Other->Pore Fluid->Fluid Leakoff**: Type: **Coefficients**: Toggle on **Use temperature-dependent data** and select the number of field variables.

Property module: material editor: **Other->Pore Fluid->Fluid Leakoff**: Type: **User**

Physics-Based Leak-Off Behavior

The physics-based leak-off behavior is based on the *Outmans (1963)* filtration models, which assume:

- the filter cake is a porous medium,
- the fluid flow through the filter cake is governed by Darcy's law, and
- the rate of increase of the thickness of the filter cake layer is proportional to the flux of fluid through the filter cake.

Abaqus assumes the filter cake to be compressible in a dynamic filtration model, while it assumes the filter cake to be incompressible in a static filtration model.

In the discussion to follow, p_f and p_p represent the (slurry) fluid pressure and formation pope fluid pressure, respectively.

For the dynamic filtration model, the porosity of the filter cake is defined as

$$\varepsilon_h = \varepsilon_0 \exp\left[-\alpha\left(p_f - p_p\right)\right],$$

where $\varepsilon_0 = C_f\varepsilon^*, \varepsilon^* = 0.68,$ is the porosity corresponding to packing of monodispersed spherical particles, C_f represents a correction factor that accounts for particle size variations, and α is a compressibility factor. The case $\alpha = 0$ represents incompressible static filter cake behavior, which implies that $\varepsilon_h = \varepsilon_0$. You specify both C_f and α to define a dynamic filtration model, or you can specify only C_f to define a static filtration model.

The porosity associated with the slurry is defined in terms of the slurry concentration as $\varepsilon_s = 1 - C_v$. The rate of growth of the thickness of the filter cake layer is assumed to be

$$\frac{dh}{dt} = b\frac{\kappa}{\mu}\frac{\partial p}{\partial x},$$

where $b = \frac{1-\varepsilon_s}{\varepsilon_s-\varepsilon_h}$ is the specific filter cake volume, p represents the fluid pressure in the filter cake, and x represents the coordinate in the thickness direction of the filter cake. You can optionally specify an initial value of the filter cake thickness, h_0. The above equation is integrated in time to obtain the thickness of the filter cake layer, $h(t)$, as a function of time. In the incompressible case, the specific filter cake volume simplifies to

$$b = \frac{C_v}{1-C_v-C_f\varepsilon^*}.$$

The permeability of the filter cake is computed as

$$\kappa = C_d\frac{\varepsilon_h}{1-\varepsilon_h},$$

where C_d is a permeability factor that you can specify directly. If you do not specify the permeability factor, Abaqus computes it internally as

$$C_d = \frac{\varepsilon_0^2 D_p^2}{150(1-\varepsilon_0)},$$

where D_p is the particle diameter. In the case of an incompressible filter cake (that is, $\alpha = 0$), the expression for permeability becomes

$$C_d = \frac{\varepsilon_0^3 D_p^2}{150(1-\varepsilon_0)^2},$$

which is very similar in form to the well-known Kozeny-Carman equation describing the permeability for a fluid flowing through a packed bed of solids.

The flux of the normal flow across the cohesive surfaces is

$$q_t = -\frac{\kappa}{\mu}\frac{(p_f-p_p)}{h(t)}\Omega,$$

where κ and μ represent the permeability of the filter cake and the filtrate viscosity, respectively; $h(t)$ is the time-dependent thickness of the filter cake layer; and $\Omega\,(\leq 1.0)$ is a numerical factor that accounts for the compressibility of the filter cake layer. For the dynamic filtration model, Abaqus computes the factor Ω internally based on α; while for the static filtration model, $\Omega = 1.0$.

Input File Usage: Use the following option to define the physics-based leak-off model:

**FLUID LEAKOFF*, SLURRY

Abaqus/CAE Usage: Defining a physics-based leak-off model is not supported in Abaqus/CAE.

User Subroutine-Based Leak-off Behavior

You can use user subroutine *UFLUIDLEAKOFF* to define a physics-based model of your choice. In this case, you define the fluid flux across the top and the bottom surfaces of the cohesive elements and their derivatives with respect to the degrees of freedom of the cohesive element directly in user subroutine *UFLUIDLEAKOFF*.

Input File Usage: Use the following option to define more complex leak-off behavior in user subroutine *UFLUIDLEAKOFF*:

**FLUID LEAKOFF*, SLURRY, USER

Abaqus/CAE Usage: Defining a user-specified leakoff model is not supported in Abaqus/CAE.

Specifying Thermal Properties of Slurry Flow

To model heat transfer associated with the slurry transport within a gap, you must define the relevant thermal properties. Optionally, the thermal properties can be defined as functions of the slurry concentration. In particular, the thermal conductivity can be defined as a function of the slurry concentration directly in a tabular form. However, the specific heat of the slurry is automatically computed based on the specific heats of the carrying fluid and the proppant particles, respectively, using a rule of mixtures.

Input File Usage: Use the following option to specify the heat capacity of the carrier fluid and the proppant particles (the heat capacities can be either constants or functions of temperature and field variables):

**SPECIFIC HEAT*, PORE FLUID, SLURRY

Use the following option to specify the thermal conductivity of the slurry (the conductivity of the slurry can be either a constant or a function of slurry concentration, temperature, and field variables):

**CONDUCTIVITY*, PORE FLUID, SLURRY

Abaqus/CAE Usage: Definitions of thermal properties of the slurry are not supported in Abaqus/CAE.

Cohesive Offset

A cohesive offset (see *Cohesive Offset*) refers to a value of the separation in the normal direction, $\delta_n = \delta_n^{off}$,

below which the resistance to additional closure $\left(\delta_n > 0, \dot{\delta}_n < 0\right)$ of the cohesive surfaces increases significantly. This capability is useful in preventing the closure of a fracture when changes in external loading conditions might otherwise result in closure. A practical application of this capability occurs during hydraulic fracturing with slurry, where solid proppant particles (the solid part of the slurry) help "prop open" the fracture when external hydraulic pressure loads are removed.

Limits on Slurry Concentration

Depending on the nature of packing of the proppant particles (assumed to be spherical), there will be a theoretical upper limit on the slurry concentration. By default, Abaqus assumes this upper limit to be 0.67. The lower limit of the slurry concentration is assumed to be 0. The strongly advective nature of the slurry transport problem can lead to oscillations in the solution; therefore, these theoretical limits might be violated numerically in some cases. By default, a box-bound constraint approach is used to enforce limits on the nodal degree of freedom representing

slurry concentration. In addition, the use of a Pertov Galerkin scheme to formulate the slurry transport continuity equation helps alleviate such numerical oscillations to a large extent but does not completely eliminate it. As discussed earlier (see *Stability of Tangential Flow*), the use of artificial diffusivity is recommended for most applications.

You can specify nondefault upper and lower limits on the slurry concentration.

Input File Usage: Use the following option to specify nondefault upper and the lower limits on the slurry concentration:

In a geostatic procedure:

**GEOSTATIC*, SLURRY MIN=*min_value*, SLURRY MAX=*max_value*

In a soils consolidation procedure:

**SOILS*, CONSOLIDATION, SLURRY MIN=*min_value*, SLURRY MAX=*max_value*

Abaqus/CAE Usage: Specifying nondefault upper and lower limits on the slurry concentration is not supported in Abaqus/CAE.

Simulating Flow of Clear Fluid

Some hydraulic fracturing applications involve the use of clear fluid to initiate fracture, followed by the use of a slurry. To model such applications, you can use multiple steps in the analysis and specify that Abaqus does not solve for slurry concentration during the step in which a clear fluid is used.

Input File Usage: Use the following option to define a flow of clear fluid:

In a geostatic procedure:

**GEOSTATIC*, SLURRY FLOW=NO

In a soils consolidation procedure:

**SOILS*, CONSOLIDATION, SLURRY FLOW=NO

Abaqus/CAE Usage: Defining a flow of clear fluid is not supported in Abaqus/CAE.

Output

In addition to the output variables that are available with fluid flow-enabled cohesive elements, the following variables are available when slurry transport is modeled:

SLURRYVF Volumetric concentration of proppant particles in the slurry.

SLURRYAF Volume of proppant particles in the slurry per unit area.

THKFTCKT Filter cake thickness with leak-off at top surface.

THKFTCKB Filter cake thickness with leak-off at bottom surface.

FLVF	When multiple fluids are defined, the components of FLVF represent the volume fraction of each fluid type. In particular, FLVF_*i* refers to the volume fraction of fluid type *i*.

References

- Outmans, H. D., "Mechanics of Static and Dynamic Filtration In the Borehole," *Society of Petroleum Engineers Journal*, vol. 3, pp. 236–244, 1963.

Two-Dimensional Cohesive Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the two-dimensional cohesive elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *About Cohesive Elements*
- *Choosing a Cohesive Element*
- **COHESIVE SECTION*
- *Adhesive joints and bonded interfaces*

Element Types

General Element

COH2D4	4-node cohesive element

Active Degrees of Freedom

1, 2

Additional Solution Variables

None.

Pore Pressure Element

COD2D4P(S)	6-node displacement and pore pressure cohesive element with the transition from Darcy flow to Poiseuille flow
COH2D4P(S)	6-node displacement and pore pressure cohesive element

Active Degrees of Freedom

1, 2, 8 at nodes on the top and bottom faces

8 at nodes on the middle face

Additional Solution Variables

None.

Coupled Temperature-Displacement Element

COH2D4T$^{(S)}$ 4-node displacement and temperature cohesive element

Active Degrees of Freedom

1, 2, 11 at nodes on the top and bottom faces

Additional Solution Variables

None.

Coupled Temperature-Pore Pressure Element

COD2D4PT$^{(S)}$ 6-node pore pressure and temperature cohesive element

Active Degrees of Freedom

1, 2, 8, 11 at nodes on the top, bottom, and middle face (1 and 2 on the middle face are constrained by corresponding values on the top and bottom faces)

Additional Solution Variables

None.

Coupled Slurry Concentration-Pore Pressure Element

COD2D4C$^{(S)}$ 6-node displacement, pore pressure and slurry concentration cohesive element

Active Degrees of Freedom

1, 2, 8 at nodes on the top and bottom faces; 1, 2, 8, 31 at nodes on the middle face (1 and 2 on the middle face are constrained by corresponding values on the top and bottom faces)

Additional Solution Variables

None.

Coupled Temperature-Slurry Concentration-Pore Pressure Element

COD2D4Q$^{(S)}$ 6-node temperature and slurry concentration cohesive element

Active Degrees of Freedom

1, 2, 8, 11 at nodes on the top and bottom faces; 1, 2, 8, 11, 31 at nodes on the middle face (1 and 2 on the middle face are constrained by corresponding values on the top and bottom faces)

Additional Solution Variables

None.

Nodal Coordinates Required

X, Y

Element Property Definition

You can define the element's initial constitutive thickness and the out-of-plane width. The default initial constitutive thickness of cohesive elements depends on the response of these elements. For continuum response, the default initial constitutive thickness is computed based on the nodal coordinates. For traction-separation response, the default initial constitutive thickness is assumed to be 1.0. For response based on a uniaxial stress state, there is no default; you must indicate your choice of the method for computing the initial constitutive thickness. See *Specifying the Constitutive Thickness* for details.

Abaqus calculates the thickness direction automatically based on the midsurface of the element.

Input File Usage: **COHESIVE SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Cohesive** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global X-direction.

Load ID (*DLOAD): BY
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global Y-direction.

Load ID (*DLOAD): BXNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *X*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BYNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *Y*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): CENT (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}(ML^{-3}T^{-2})$

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): CENTRIF (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): CORIO (S)
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-4}T$ $(ML^{-3}T^{-1})$

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): P*n*
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on face *n*.

Load ID (*DLOAD): P*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure on face *n* with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): ROTA (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}
Description: Rotary acceleration load (magnitude is input as α, where α is the rotary acceleration).

Load ID (*DLOAD): SBF (E)
Abaqus/CAE Load/Interaction: Not supported

Units: $FL^{-5}T^2$

Description: Stagnation body force in global *X*- and *Y*-directions.

Load ID (*DLOAD): SP*n* (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T^2$
Description: Stagnation pressure on face *n*.

Load ID (*DLOAD): VBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T$
Description: Viscous body force in global *X*- and *Y*-directions.

Load ID (*DLOAD): VP*n* (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-3}T$
Description: Viscous pressure on face *n*, applying a pressure proportional to the velocity normal to the face and opposing the motion.

Distributed Heat Fluxes

Distributed heat fluxes are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DFLUX): BF
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Heat body flux per unit volume.

Load ID (*DFLUX): BFNU
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Nonuniform heat body flux per unit volume with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard.

Load ID (*DFLUX): S*n*
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Heat surface flux per unit area into face *n*.

Load ID (*DFLUX): S*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}$
Description: Nonuniform heat surface flux per unit area into face *n* with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard.

Film Conditions

Film conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*FILM): F*n*
Abaqus/CAE Load/Interaction: Surface film condition

Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Film coefficient and sink temperature (units of θ) provided on face *n*.

Load ID (*FILM): F*n*NU (S)

Abaqus/CAE Load/Interaction: Not supported

Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of θ) provided on face *n* with magnitude supplied via user subroutine *FILM*.

Radiation Types

Radiation conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*RADIATE): R*n*

Abaqus/CAE Load/Interaction: Surface radiation

Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) provided on face *n*.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): P

Abaqus/CAE Load/Interaction: Pressure

Units: FL^{-2}

Description: Pressure on the element surface.

Load ID (*DSLOAD): PNU

Abaqus/CAE Load/Interaction: Pressure

Units: FL^{-2}

Description: Nonuniform pressure on the element surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DSLOAD): SP (E)

Abaqus/CAE Load/Interaction: Pressure

Units: $FL^{-4}T^{2}$

Description: Stagnation pressure on the element surface.

Load ID (*DSLOAD): VP (E)

Abaqus/CAE Load/Interaction: Pressure

Units: $FL^{-3}T$

Description: Viscous pressure applied on the element surface. The viscous pressure is proportional to the velocity normal to the element face and opposing the motion.

Element Output

Stress, strain, and other tensor components available for output depend on whether the cohesive elements are used to model adhesive joints, gaskets, or delamination problems. You indicate the intended usage of the cohesive

elements by choosing an appropriate response type when defining the section properties of these elements. The available response types are discussed in *Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach* and *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*.

Cohesive Elements Using a Continuum Response

Stress and other tensors (including strain tensors) are available for elements with continuum response. Both the stress tensor and the strain tensor contain true values. For the constitutive calculations using a continuum response, only the direct through-thickness and the transverse shear strains are assumed to be nonzero. All the other strain components (that is, the membrane strains) are assumed to be zero (see *Modeling of an Adhesive Layer of Finite Thickness* for details). All tensors have the same number of components. For example, the stress components are as follows:

S11 Direct membrane stress.

S22 Direct through-thickness stress.

S33 Direct membrane stress.

S12 Transverse shear stress.

Cohesive Elements Using a Uniaxial Stress State

Stress and other tensors (including strain tensors) are available for cohesive elements with uniaxial stress response. Both the stress tensor and the strain tensor contain true values. For the constitutive calculations using a uniaxial stress response, only the direct through-thickness stress is assumed to be nonzero. All the other stress components (that is, the membrane and transverse shear stresses) are assumed to be zero (see *Modeling of Gaskets and/or Small Adhesive Patches* for details). All tensors have the same number of components. For example, the stress components are as follows:

S22 Direct through-thickness stress.

Cohesive Elements Using a Traction-Separation Response

Stress and other tensors (including strain tensors) are available for elements with traction-separation response. Both the stress tensor and the strain tensor contain nominal values. The output variables E, LE, and NE all contain the nominal strain when the response of cohesive elements is defined in terms of traction versus separation. The output variable ER contains the nominal strain rate. All tensors have the same number of components. For example, the stress components are as follows:

S22 Direct through-thickness stress.

S12 Transverse shear stress.

Node Ordering and Face Numbering on Elements

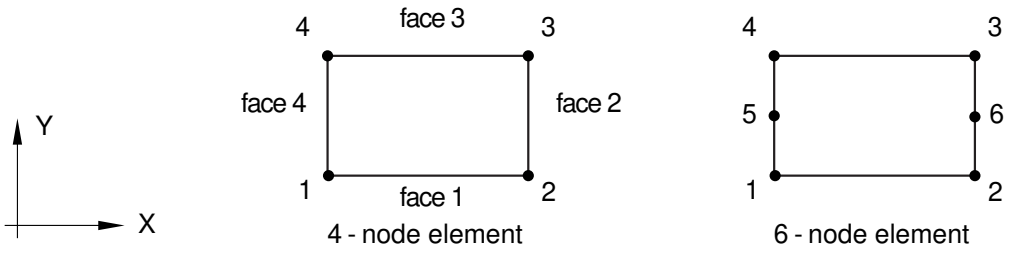
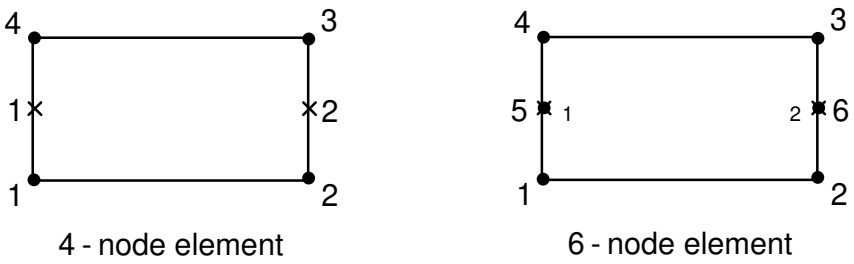


Table 1: Element faces

Face 1	1 – 2 face
Face 2	2 – 3 face
Face 3	3 – 4 face
Face 4	4 – 1 face

Numbering of Integration Points for Output



Three-Dimensional Cohesive Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the three-dimensional cohesive elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *About Cohesive Elements*
- *Choosing a Cohesive Element*
- **COHESIVE SECTION*
- *Adhesive joints and bonded interfaces*

Element Types

General Elements

COH3D6	6-node cohesive element
COH3D8	8-node cohesive element

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

None.

Pore Pressure Elements

COD3D6P$^{(S)}$	9-node displacement and pore pressure cohesive element with the transition from Darcy flow to Poiseuille flow
COD3D8P$^{(S)}$	12-node displacement and pore pressure cohesive element with the transition from Darcy flow to Poiseuille flow
COH3D6P$^{(S)}$	9-node displacement and pore pressure cohesive element
COH3D8P$^{(S)}$	12-node displacement and pore pressure cohesive element

Active Degrees of Freedom

1, 2, 3, 8 at nodes on the top and bottom faces

8 at nodes on the middle face

Additional Solution Variables

None.

Coupled Temperature-Displacement Elements

COH3D6T$^{(S)}$	6-node displacement and temperature cohesive element
COH3D8T$^{(S)}$	8-node displacement and temperature cohesive element

Active Degrees of Freedom

1, 2, 3, 11 at nodes on the top and bottom faces

Additional Solution Variables

None.

Coupled Temperature-Pore Pressure Elements

COD3D6PT$^{(S)}$	9-node temperature and pore pressure cohesive element
COD3D8PT$^{(S)}$	12-node temperature and pore pressure cohesive element

Active Degrees of Freedom

1, 2, 3, 8, 11 at nodes on the top, bottom, and middle face (1, 2, and 3 on the middle face are constrained by corresponding values on the top and bottom faces)

Additional Solution Variables

None.

Coupled Slurry Concentration-Pore Pressure Elements

COD3D6C$^{(S)}$	9-node displacement, pore pressure, and slurry concentration cohesive element
COD3D8C$^{(S)}$	12-node displacement, pore pressure, and slurry concentration cohesive element

Active Degrees of Freedom

1, 2, 3, 8 at nodes on the top and bottom faces; 1, 2, 3, 8, 31 at nodes on the middle face (1, 2, and 3 on the middle face are constrained by corresponding values on the top and bottom faces)

Additional Solution Variables

None.

Coupled Temperature-Slurry Concentration-Pore Pressure Elements

COD3D6Q$^{(S)}$	9-node temperature and slurry concentration cohesive element
COD3D8Q$^{(S)}$	12-node temperature and slurry concentration cohesive element

Active Degrees of Freedom

1, 2, 3, 8, 11 at nodes on the top, bottom, and middle face (1, 2, and 3 on the middle face are constrained by corresponding values on the top and bottom faces)

Additional Solution Variables

None.

Nodal Coordinates Required

X, Y, Z

Element Property Definition

You can define the element's initial constitutive thickness. The default initial constitutive thickness of cohesive elements depends on the response of these elements. For continuum response, the default initial constitutive thickness is computed based on the nodal coordinates. For traction-separation response, the default initial constitutive thickness is assumed to be 1.0. For response based on a uniaxial stress state, there is no default; you must indicate your choice of the method for computing the initial constitutive thickness. See *Specifying the Constitutive Thickness* for details.

Abaqus computes the thickness direction automatically based on the midsurface of the element.

Input File Usage: **COHESIVE SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Cohesive** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}

Description: Body force in global *X*-direction.

Load ID (*DLOAD): BY
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *Y*-direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *Z*-direction.

Load ID (*DLOAD): BXNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *X*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BYNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *Y*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *Z*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): CENT (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}(ML^{-3}T^{-2})$

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): CENTRIF (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): CORIO (S)
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-4}T$ $(ML^{-3}T^{-1})$

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): P*n*

Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on face *n*.

Load ID (*DLOAD): P*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure on face *n* with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): ROTA (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}
Description: Rotary acceleration load (magnitude is input as $\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is the rotary acceleration).

Load ID (*DLOAD): SBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-5}T^2$
Description: Stagnation body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): SP*n* (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T^2$
Description: Stagnation pressure on face *n*.

Load ID (*DLOAD): VBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T$
Description: Viscous body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): VP*n* (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-3}T$
Description: Viscous pressure on face *n*, applying a pressure proportional to the velocity normal to the face and opposing the motion.

Distributed Heat Fluxes

Distributed heat fluxes are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*DFLUX): BF
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Heat body flux per unit volume.

Load ID (*DFLUX): BFNU
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Nonuniform heat body flux per unit volume with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard.

Load ID (*DFLUX): MBFNU (S)

Abaqus/CAE Load/Interaction: Not supported
Units: JT^{-1}

Description: Nonuniform moving or stationary concentrated heat fluxes with magnitudes supplied via user subroutine *UMDFLUX*.

Load ID (*DFLUX): S*n*
Abaqus/CAE Load/Interaction: **Surface heat flux**
Units: $JL^{-2}T^{-1}$

Description: Heat surface flux per unit area into face *n*.

Load ID (*DFLUX): S*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}$

Description: Nonuniform heat surface flux per unit area into face *n* with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard.

Film Conditions

Film conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*FILM): F*n*
Abaqus/CAE Load/Interaction: **Surface film condition**
Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Film coefficient and sink temperature (units of θ) provided on face *n*.

Load ID (*FILM): F*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of θ) provided on face *n* with magnitude supplied via user subroutine *FILM*.

Load ID (*FILM): FFS (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Film coefficient and sink temperature (units of θ) provided on all free faces of an element.

Load ID (*FILM): FFSNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\theta^{-1}$

Description: Nonuniform film coefficient and sink temperature (units of θ) provided on all free faces of an element with magnitude supplied via user subroutine *FILM*.

Radiation Types

Radiation conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*RADIATE): R*n*
Abaqus/CAE Load/Interaction: **Surface radiation**
Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) provided on face n.

Load ID (*RADIATE): RFS
Abaqus/CAE Load/Interaction: Not supported
Units: Dimensionless

Description: Emissivity and sink temperature (units of θ) provided on free faces of an element.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element surface.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Nonuniform pressure on the element surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DSLOAD): SP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-4}T^2$
Description: Stagnation pressure on the element surface.

Load ID (*DSLOAD): VP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-3}T$
Description: Viscous pressure applied on the element surface. The viscous pressure is proportional to the velocity normal to the element face and opposing the motion.

Element Output

Stress, strain, and other tensor components available for output depend on whether the cohesive elements are used to model adhesive joints, gaskets, or delamination problems. You indicate the intended usage of the cohesive elements by choosing an appropriate response type when defining the section properties of these elements. The available response types are discussed in *Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach* and *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*.

Cohesive Elements Using a Continuum Response

Stress and other tensors (including strain tensors) are available for elements with continuum response. Both the stress tensor and the strain tensor contain true values. For the constitutive calculations using a continuum response, only the direct through-thickness and the transverse shear strains are assumed to be nonzero. All the other strain components (that is, the membrane strains) are assumed to be zero (see *Modeling of an Adhesive Layer of Finite Thickness* for details). All tensors have the same number of components. For example, the stress components are as follows:

S11 Direct membrane stress.

S22 Direct membrane stress.

S33 Direct through-thickness stress.

S12 In-plane membrane shear stress.

S13 Transverse shear stress.

S23 Transverse shear stress.

Cohesive Elements Using a Uniaxial Stress State

Stress and other tensors (including strain tensors) are available for cohesive elements with uniaxial stress response. Both the stress tensor and the strain tensor contain true values. For the constitutive calculations using a uniaxial stress response, only the direct through-thickness stress is assumed to be nonzero. All the other stress components (that is, the membrane and transverse shear stresses) are assumed to be zero (see *Modeling of Gaskets and/or Small Adhesive Patches* for details). All tensors have the same number of components. For example, the stress components are as follows:

S33 Direct through-thickness stress.

Cohesive Elements Using a Traction-Separation Response

Stress and other tensors (including strain tensors) are available for elements with traction-separation response. Both the stress tensor and the strain tensor contain nominal values. The output variables E, LE, and NE all contain the nominal strain when the response of cohesive elements is defined in terms of traction versus separation. The output variable ER contains the nominal strain rate. All tensors have the same number of components. For example, the stress components are as follows:

S33 Direct through-thickness stress.

S13 Transverse shear stress.

S23 Transverse shear stress.

Node Ordering and Face Numbering on Elements

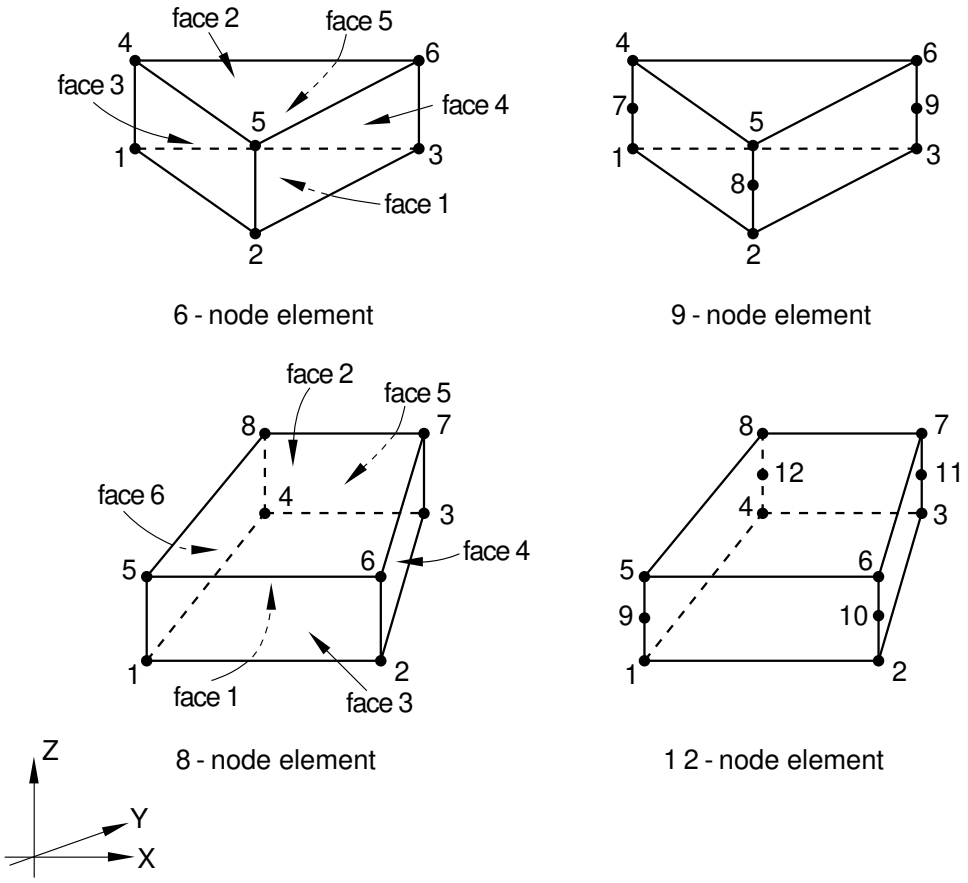


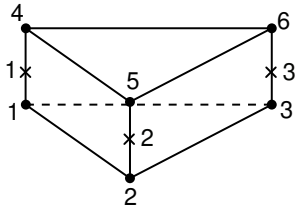
Table 1: Element faces for COH3D6 and COD3D6P

Face 1	1 – 2 – 3 face
Face 2	4 – 6 – 5 face
Face 3	1 – 4 – 5 – 2 face
Face 4	2 – 5 – 6 – 3 face
Face 5	3 – 6 – 4 – 1 face

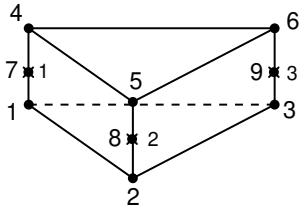
Table 2: Element faces for COH3D8 and COD3D8P

Face 1	1 – 2 – 3 – 4 face
Face 2	5 – 8 – 7 – 6 face
Face 3	1 – 5 – 6 – 2 face
Face 4	2 – 6 – 7 – 3 face
Face 5	3 – 7 – 8 – 4 face
Face 6	4 – 8 – 5 – 1 face

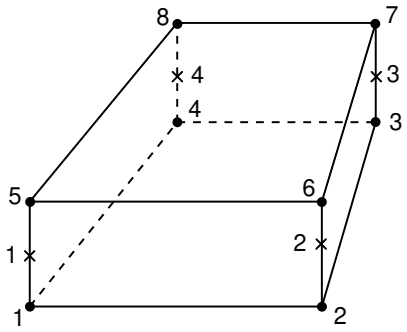
Numbering of Integration Points for Output



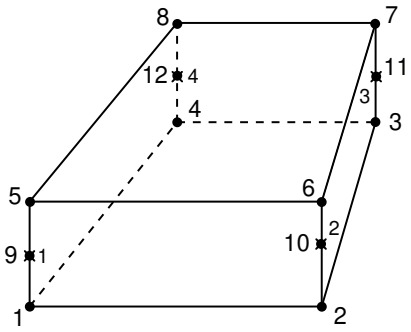
6 - node element



9 - node element



8 - node element



1 2 - node element

Axisymmetric Cohesive Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the axisymmetric cohesive elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *About Cohesive Elements*
- *Choosing a Cohesive Element*
- **COHESIVE SECTION*
- *Adhesive joints and bonded interfaces*

Element Types

General Element

COHAX4 4-node cohesive element

Active Degrees of Freedom

1, 2 (u_r, u_z)

Additional Solution Variables

None.

Pore Pressure Elements

CODAX4P$^{(S)}$ 6-node displacement and pore pressure cohesive element with the transition from Darcy flow to Poiseuille flow

COHAX4P$^{(S)}$ 6-node displacement and pore pressure cohesive element

Active Degrees of Freedom

1, 2, 8

Additional Solution Variables

None.

Coupled Temperature-Displacement Element

COHAX4T$^{(S)}$ 4-node displacement and temperature cohesive element

Active Degrees of Freedom

1, 2, 11

Additional Solution Variables

None.

Coupled Temperature-Pore Pressure Element

CODAX4PT$^{(S)}$ 6-node temperature and pore pressure cohesive element

Active Degrees of Freedom

1, 2, 8,11 (1 and 2 on the middle face are constrained by corresponding values on the top and bottom faces)

Additional Solution Variables

None.

Coupled Slurry Concentration-Pore Pressure Element

CODAX4C$^{(S)}$ 6-node displacement, pore pressure, and slurry concentration cohesive element

Active Degrees of Freedom

1, 2, 8 at nodes on the top and bottom faces; 1, 2, 8, 31 at nodes on the middle face (1 and 2 on the middle face are constrained by corresponding values on the top and bottom faces)

Additional Solution Variables

None.

Coupled Temperature-Slurry Concentration-Pore Pressure Element

CODAX4Q$^{(S)}$ 6-node temperature and slurry cohesive element

Active Degrees of Freedom

1, 2, 8, 11 at nodes on the top and bottom faces; 1, 2, 8, 11, 31 at nodes on the middle face (1 and 2 on the middle face are constrained by corresponding values on the top and bottom faces)

Additional Solution Variables

None.

Nodal Coordinates Required

X, Y

Element Property Definition

You can define the element's initial constitutive thickness. The default initial constitutive thickness of cohesive elements depends on the response of these elements. For continuum response, the default initial constitutive thickness is computed based on the nodal coordinates. For traction-separation response, the default initial constitutive thickness is assumed to be 1.0. For response based on a uniaxial stress state, there is no default; you must indicate your choice of the method for computing the initial constitutive thickness. See *Specifying the Constitutive Thickness* for details.

Abaqus calculates the thickness direction automatically based on the midsurface of the element.

Input File Usage: **COHESIVE SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Cohesive** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BR
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in radial direction.

Load ID (*DLOAD): BY
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in axial direction.

Load ID (*DLOAD): BRNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}

Description: Nonuniform body force in radial direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in axial direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): CENT (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}(ML^{-3}T^{-2})$

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume, ω is the angular velocity).

Load ID (*DLOAD): CENTRIF (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity).

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): P*n*
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on face *n*.

Load ID (*DLOAD): P*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure on face *n* with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): SBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-5}T^2$
Description: Stagnation body force in radial and axial directions.

Load ID (*DLOAD): SP*n* (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T^2$
Description: Stagnation pressure on face *n*.

Load ID (*DLOAD): VBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T$
Description: Viscous body force in radial and axial directions.

Load ID (*DLOAD): VP*n* (E)
Abaqus/CAE Load/Interaction: Not supported

Units: $FL^{-3}T$

Description: Viscous pressure on face *n*, applying a pressure proportional to the velocity normal to the face and opposing the motion.

Distributed Heat Fluxes

Distributed heat fluxes are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*. Distributed heat flux magnitudes are per unit area or per unit volume. They do not need to be multiplied by 2π.

Load ID (*DFLUX): BF
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Heat body flux per unit volume.

Load ID (*DFLUX): BFNU
Abaqus/CAE Load/Interaction: Body heat flux
Units: $JL^{-3}T^{-1}$
Description: Nonuniform heat body flux per unit volume with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard.

Load ID (*DFLUX): S*n*
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Heat surface flux per unit area into face *n*.

Load ID (*DFLUX): S*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}$
Description: Nonuniform heat surface flux per unit area into face *n* with magnitude supplied via user subroutine *DFLUX* in Abaqus/Standard.

Film Conditions

Film conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*FILM): F*n*
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\theta^{-1}$
Description: Film coefficient and sink temperature (units of θ) provided on face *n*.

Load ID (*FILM): F*n*NU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: $JL^{-2}T^{-1}\theta^{-1}$
Description: Nonuniform film coefficient and sink temperature (units of θ) provided on face *n* with magnitude supplied via user subroutine *FILM*.

Radiation Types

Radiation conditions are available for all elements with temperature degrees of freedom. They are specified as described in *Thermal Loads*.

Load ID (*RADIATE): R*n*
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless
Description: Emissivity and sink temperature provided for face *n*.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element surface.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Nonuniform pressure on the element surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DSLOAD): SP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-4}T^2$
Description: Stagnation pressure on the element surface.

Load ID (*DSLOAD): VP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-3}T$
Description: Viscous pressure applied on the element surface. The viscous pressure is proportional to the velocity normal to the element face and opposing the motion.

Element Output

Stress, strain, and other tensor components available for output depend on whether the cohesive elements are used to model adhesive joints, gaskets, or delamination problems. You indicate the intended usage of the cohesive elements by choosing an appropriate response type when defining the section properties of these elements. The available response types are discussed in *Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach* and *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*.

Cohesive Elements Using a Continuum Response

Stress and other tensors (including strain tensors) are available for elements with continuum response. Both the stress tensor and the strain tensor contain true values. For the constitutive calculations using a continuum response, only the direct through-thickness and the transverse shear strains are assumed to be nonzero. All the other strain components (that is, the membrane strains) are assumed to be zero (see *Modeling of an Adhesive Layer of Finite Thickness* for details). All tensors have the same number of components. For example, the stress components are as follows:

S11 Direct membrane stress.

S22 Direct through-thickness stress.

S33 Direct membrane stress.

S12 Transverse shear stress.

Cohesive Elements Using a Uniaxial Stress State

Stress and other tensors (including strain tensors) are available for cohesive elements with uniaxial stress response. Both the stress tensor and the strain tensor contain true values. For the constitutive calculations using a uniaxial stress response, only the direct through-thickness stress is assumed to be nonzero. All the other stress components (that is, the membrane and transverse shear stresses) are assumed to be zero (see *Modeling of Gaskets and/or Small Adhesive Patches* for details). All tensors have the same number of components. For example, the stress components are as follows:

S22 Direct through-thickness stress.

Cohesive Elements Using a Traction-Separation Response

Stress and other tensors (including strain tensors) are available for elements with traction-separation response. Both the stress tensor and the strain tensor contain nominal values. The output variables E, LE, and NE all contain the nominal strain when the response of cohesive elements is defined in terms of traction versus separation. All tensors have the same number of components. For example, the stress components are as follows:

S22 Direct through-thickness stress.

S12 Transverse shear stress.

Node Ordering and Face Numbering on Elements

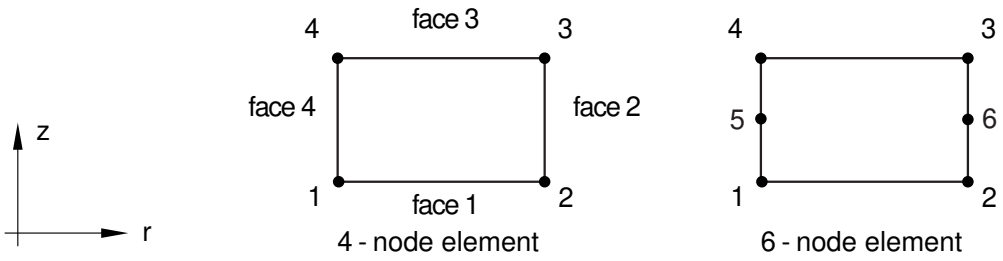
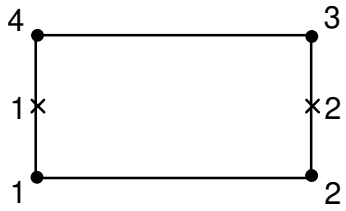


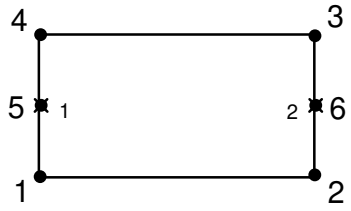
Table 1: Element faces

Face 1	1 – 2 face
Face 2	2 – 3 face
Face 3	3 – 4 face
Face 4	4 – 1 face

Numbering of Integration Points for Output



4 - node element



6 - node element

Dashpot Elements

In this section:

- *Dashpots*
- *Dashpot Element Library*

Dashpots

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Dashpot Element Library*
- **DASHPOT*
- *Defining springs and dashpots*

Overview

Dashpot elements:

- can couple a force with a relative velocity;
- in Abaqus/Standard can couple a moment with a relative angular velocity;
- can be linear or nonlinear;
- if linear, can be dependent on frequency in direct-solution steady-state dynamic analysis;
- can be dependent on temperature and field variables; and
- can be used in any stress analysis procedure.

The terms “force” and “velocity” are used throughout the description of dashpot elements. When the dashpot is associated with displacement degrees of freedom, these variables are the force and relative velocity in the dashpot. If the dashpots are associated with rotational degrees of freedom, they are torsional dashpots; these variables will then be the moment transmitted by the dashpot and the relative angular velocity across the dashpot.

In dynamic analysis the velocities are obtained as part of the integration operator; in quasi-static analysis in Abaqus/Standard the velocities are obtained by dividing the displacement increments by the time increment.

Typical Applications

Dashpots are used to model relative velocity-dependent force or torsional resistance. They can also provide viscous energy dissipation mechanisms.

Dashpots are often useful in unstable, nonlinear, static analyses where the modified Riks algorithm is not appropriate (see *Unstable Collapse and Postbuckling Analysis* for a discussion of the modified Riks algorithm) and where the automatic time stepping algorithm is used because sudden shifts in configuration can be controlled by the forces that arise in the dashpots. In such cases the magnitude of the damping must be chosen in conjunction with the time period so that enough damping is available to control such difficulties but the damping forces are negligible when a stable static response is obtained. See also the contact damping available with contact elements in Abaqus/Standard (see *Contact Damping*).

Choosing an Appropriate Element

DASHPOT1 and DASHPOT2 elements are available only in Abaqus/Standard. DASHPOT1 is between a specified degree of freedom and ground. DASHPOT2 is between two specified degrees of freedom.

The DASHPOTA element is available in both Abaqus/Standard and Abaqus/Explicit. DASHPOTA is between two nodes with its line of action being the line joining the two nodes.

The dashpot behavior can be linear or nonlinear in any of these elements.

Input File Usage: Use the following option to specify a dashpot element between a specified degree of freedom and ground:

**ELEMENT*, TYPE=DASHPOT1

Use the following option to specify a dashpot element between two degrees of freedom:

**ELEMENT*, TYPE=DASHPOT2

Use the following option to specify a dashpot element between two nodes with its line of action being the line joining the two nodes:

**ELEMENT*, TYPE=DASHPOTA

Abaqus/CAE Usage: Property or Interaction module: **Special->Springs/Dashpots->Create**, then select one of the following:
Connect points to ground: select points: toggle on **Dashpot coefficient** (*equivalent to DASHPOT1*)
Connect two points: select points: **Axis**: **Specify fixed direction**: toggle on **Dashpot coefficient** (*equivalent to DASHPOT2*)
Connect two points: select points: **Axis**: **Follow line of action**: toggle on **Dashpot coefficient** (*equivalent to DASHPOTA*)

Stability Considerations in Abaqus/Explicit

Abaqus/Explicit does not take dashpots into account when determining the stable time step; therefore, care should be taken when introducing dashpots into the mesh.

A DASHPOTA element introduces a damping force between two degrees of freedom without introducing any stiffness between these degrees of freedom and without introducing any mass at the nodes. This can cause a reduction in the stable time increment. For example, consider a simple system of a truss element and a dashpot element as shown in *Figure 1*.

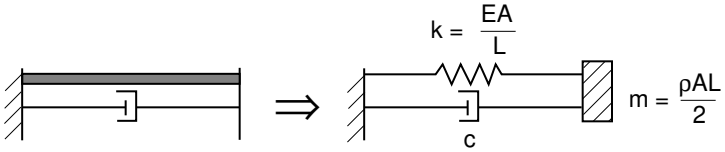


Figure 1: A simple truss and dashpot system.

The dynamic equation for this system is

$$m\ddot{x} + c\dot{x} + kx = 0$$

or

$$\ddot{x} + 2\xi w\dot{x} + w^2 x = 0,$$

where

$$w^2 = \frac{k}{m}$$

and

$$\xi = \frac{c}{2\sqrt{mk}}.$$

The stable time increment for the spring-dashpot system is

$$\Delta t_{\text{stable}} = \frac{2}{w}\left(\sqrt{1+\xi^2} - \xi\right).$$

As the dashpot coefficient c is increased, the stable time increment, Δt_{stable}, will be reduced.

To avoid this reduction in the stable time increment, dashpots should be used in parallel with spring or truss elements, where the stiffness of the spring or truss elements is chosen so that the stable time increment of the dashpot and spring or truss is larger than the stable critical time increment that is calculated by Abaqus/Explicit. If this requires springs or trusses that have unacceptable forces, specify the time increment size directly for the step (see *Explicit Dynamic Analysis*).

Relative Velocity Definition

The relative velocity definition depends on the element type.

DASHPOT1 Elements

The relative velocity across a DASHPOT1 element is the *i*th component of velocity of the dashpot's node:

$$\Delta v = v_i,$$

where *i* is defined as described below and can be in a local direction (see *Defining the Direction of Action for DASHPOT1 and DASHPOT2 Elements*).

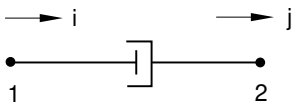
DASHPOT2 Elements

The relative velocity across a DASHPOT2 element is the difference between the *i*th component of velocity at the dashpot's first node and the *j*th component of velocity of the dashpot's second node:

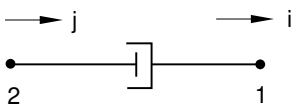
$$\Delta v = v_i^1 - v_j^2,$$

where *i* and *j* are defined as described below and can be in local directions (see *Defining the Direction of Action for DASHPOT1 and DASHPOT2 Elements*).

It is important to understand how the DASHPOT2 element will behave according to the above relative displacement equation since the element can produce counterintuitive results. For example, a DASHPOT2 element set up in the following way will be a “compressive” dashpot:



If the nodes have velocities such that $v_i^1 = 1$ and $v_j^2 = 0$, the dashpot is compressed while the force in the dashpot is positive. To obtain a “tensile” dashpot, the DASHPOT2 element should be set up in the following way:



DASHPOTA Elements

The relative velocity across a DASHPOTA element is the difference between the velocity of the dashpot's second node and the dashpot's first node, taken in the direction of the current axis of the dashpot.

For geometrically linear analysis,

$$\Delta v = \left(\mathbf{v}^2 - \mathbf{v}^1\right) \cdot \frac{\mathbf{X}^2 - \mathbf{X}^1}{l_0},$$

where $\mathbf{X}^1$ is the reference position of the dashpot's first node, $\mathbf{X}^2$ is the reference position of the dashpot's second node, and l_0 is the reference length of the dashpot.

For geometrically nonlinear analysis,

$$\Delta v = \left(\mathbf{v}^2 - \mathbf{v}^1\right) \cdot \frac{\mathbf{x}^2 - \mathbf{x}^1}{l},$$

where $\mathbf{x}^1$ is the current position of the dashpot's first node, $\mathbf{x}^2$ is the current position of the dashpot's second node, and l is the current length of the dashpot.

In either case the force in a DASHPOTA element is positive if the dashpot is extending.

Defining Dashpot Behavior

The dashpot behavior can be linear or nonlinear. In either case you must associate the dashpot behavior with a region of your model.

Input File Usage: **DASHPOT*, ELSET=*name*

where the ELSET parameter refers to a set of dashpot elements.

Abaqus/CAE Usage: Property or Interaction module: **Special->Springs/Dashpots->Create**: select connectivity type: select points

Linear Dashpot Behavior

You define linear dashpot behavior by specifying a constant dashpot coefficient (force per relative velocity).

The dashpot coefficient can depend on temperature and field variables. See *Input Syntax Rules* for further information about defining data as functions of temperature and independent field variables.

For direct-solution steady-state dynamic analysis the dashpot coefficient can depend on frequency, as well as on temperature and field variables. If a frequency-dependent dashpot coefficient is specified for any other analysis procedure in Abaqus/Standard, the data for the lowest frequency given will be used.

Input File Usage: **DASHPOT*, DEPENDENCIES=*n*
first data line
dashpot coefficient, *frequency*, *temperature*, *field variable 1*, etc.
...

Abaqus/CAE Usage: Property or Interaction module: **Special->Springs/Dashpots->Create**: select connectivity type: select points: **Property**: **Dashpot coefficient**: *dashpot coefficient*

Defining the dashpot coefficient as a function of frequency, temperature, and field variables is not supported in Abaqus/CAE when you define dashpots as engineering

features; instead, you can define connectors that have dashpot-like damping behavior (see *Connector Damping Behavior*).

Nonlinear Dashpot Behavior

You define nonlinear dashpot behavior by giving pairs of force–relative velocity values. These values should be given in ascending order of relative velocity and should be provided over a sufficiently wide range of relative velocity values so that the behavior is defined correctly. Abaqus assumes that the force remains constant outside the range given (see *Figure 2*). In addition, the curve should pass through the origin. That is, the force should be zero at zero relative velocity.

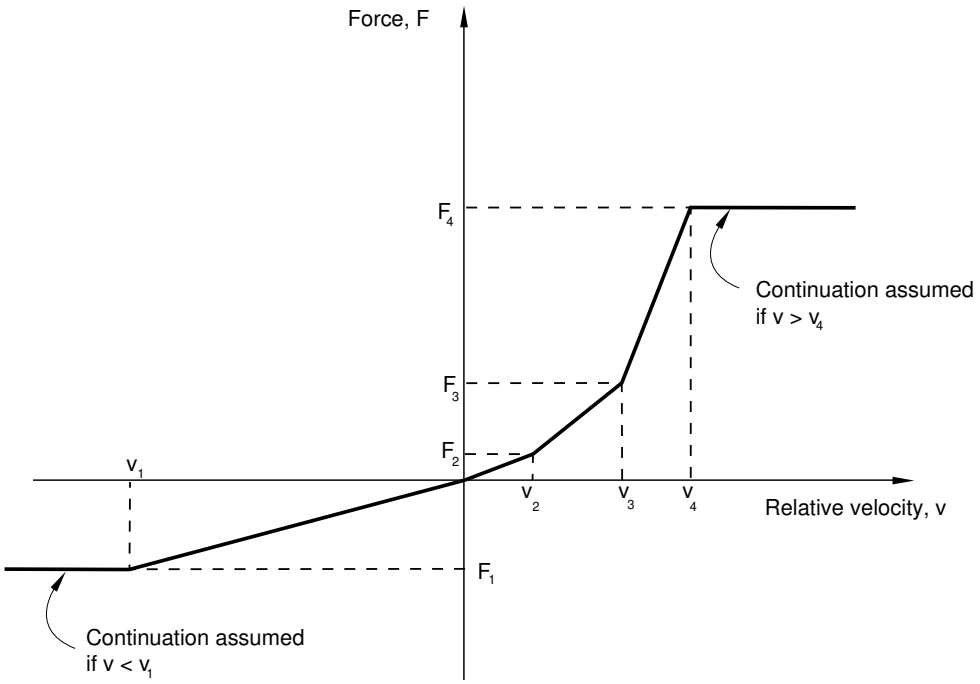


Figure 2: Nonlinear dashpot force-relative velocity relationship.

The dashpot coefficient can depend on temperature and field variables. See *Input Syntax Rules* for further information about defining data as functions of temperature and independent field variables.

Abaqus/Explicit will regularize the data into tables that are defined in terms of even intervals of the independent variables. In some cases where the force is defined at uneven intervals of the independent variable (relative velocity) and the range of the independent variable is large compared to the smallest interval, Abaqus/Explicit may fail to obtain an accurate regularization of your data in a reasonable number of intervals. In this case the program will stop after all data are processed with an error message that you must redefine the material data. See *Material Data Definition* for a more detailed discussion of data regularization.

Input File Usage:

**DASHPOT*, NONLINEAR, DEPENDENCIES=*n*
first data line
force, *relative velocity*, *temperature*, *field variable 1*, etc.
...

Abaqus/CAE Usage:

Defining nonlinear dashpot behavior is not supported in Abaqus/CAE when you define dashpots as engineering features; instead, you can define connectors that have dashpot-like damping behavior (see *Connector Damping Behavior*).

Defining the Direction of Action for DASHPOT1 and DASHPOT2 Elements

You define the direction of action for DASHPOT1 and DASHPOT2 elements by giving the degree of freedom at each node of the element. This degree of freedom may be in a local coordinate system (*Orientations*). This local system is assumed to be fixed: even in large-displacement analysis DASHPOT1 and DASHPOT2 elements act in a fixed direction throughout the analysis.

Input File Usage: **DASHPOT*, ORIENTATION=*name*
dof at node 1, *dof at node 2*

Abaqus/CAE Usage: Property or Interaction module: **Special->Springs/Dashpots->Create**, then select one of the following:
Connect points to ground: select points: **Orientation**: **Edit**: select orientation
Connect two points: select points: **Axis**: **Specify fixed direction**: **Orientation**: **Edit**: select orientation

Dashpots within Substructures

Dashpots cannot be used within substructures. You can define Rayleigh damping within the substructure definition or on the usage level to create damping within a substructure; see *Defining Substructure Damping* for more information.

Dashpot Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the dashpot elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *Dashpots*
- **DASHPOT*

Element Types

DASHPOTA	Axial dashpot between two nodes, whose line of action is the line joining the two nodes
DASHPOT1$^{(S)}$	Dashpot between a node and ground, acting in a fixed direction
DASHPOT2$^{(S)}$	Dashpot between two nodes, acting in a fixed direction

Active Degrees of Freedom

DASHPOTA: 1, 2, 3. The translational degree of freedom in the 3-direction is not activated in an Abaqus/Standard analysis if both nodes of the element are connected to two-dimensional entities such as two-dimensional analytical rigid surfaces, two-dimensional beam elements, etc.

DASHPOT1 or DASHPOT2: 1, 2, 3, 4, 5, or 6. If you specify a local orientation for the dashpot, these are local degrees of freedom. Otherwise, these are global degrees of freedom.

Additional Solution Variables

None.

Nodal Coordinates Required

DASHPOTA: *X*, *Y*, *Z*. These coordinates are used in the calculation of the action of the element.

DASHPOT1 or DASHPOT2: None. The element nodes do not need to have coordinates defined since the action associated with these elements is defined by specifying the degrees of freedom involved.

Element Property Definition

Input File Usage: **DASHPOT*

Abaqus/CAE Usage: Property or Interaction module: **Special->Springs/Dashpots->Create**

Element-Based Loading

None.

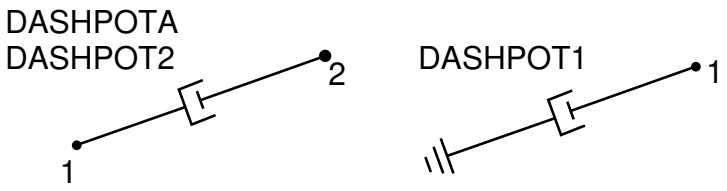
Element Output

S11 The force in the dashpot.

E11 The relative displacement across the dashpot.

ER11 The relative velocity across the dashpot (available only from Abaqus/Standard).

Node Ordering on Elements



Distributing Coupling Elements

In this section:

- *Distributing Coupling Elements*
- *Distributing Coupling Element Library*

Distributing Coupling Elements

Products: Abaqus/Standard

References:

- *Distributing Coupling Element Library*
- **DISTRIBUTING COUPLING*

Overview

Distributing coupling elements:

- can be used to distribute forces and moments on a reference node to a collection of nodes;
- can be used to prescribe an average displacement and rotation to a collection of nodes;
- can control the force distribution through the use of weight factors specified for each coupling node;
- can be used to create a flexible coupling between structural and solid elements; and
- can be used with two- or three-dimensional stress/displacement elements.

The preferred method for defining a distributing constraint is described in *Coupling Constraints*.

Typical Applications

The distributing coupling element constrains the motion of the coupling nodes to the translation and rotation of the element node. This constraint is enforced in an average sense and in a way that enables control of the transmission of loads. These characteristics make the distributing coupling element useful in a number of applications:

- The element can be used to prescribe a displacement and rotation condition on a boundary in cases where relative motion among the nodes on the boundary is required. An example of such a case is prescribing a twist on the end of a structure that is expected to warp and/or deform within the end surface (see *Figure 1*).

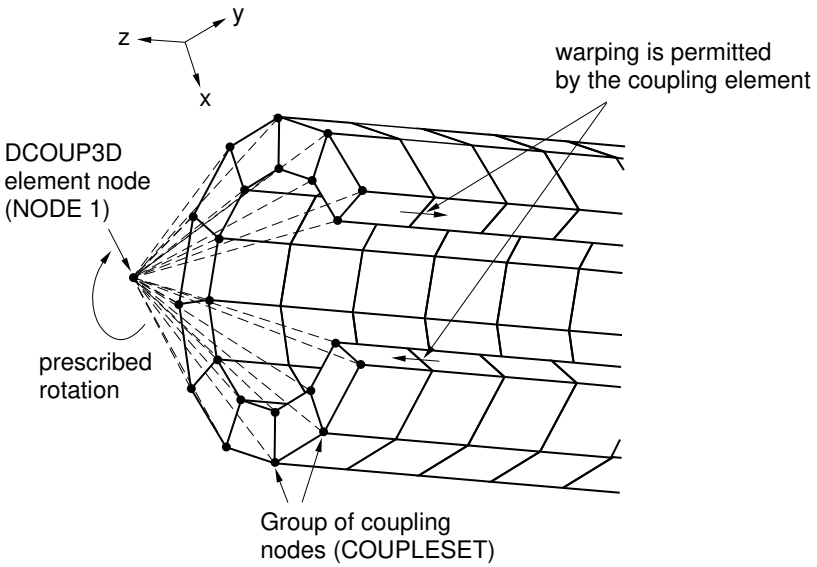


Figure 1: DCOUP3D element used to impart a rotation on the surface of a structure without constraining motion within the surface.

- The element can be used to provide, through the motion of the reference node, a weighted average of the motion of the coupling nodes.
- The element can be used to distribute loads, where the load distribution can be described with moment-of-inertia expressions. Examples of such cases include the classic bolt-pattern and weld-pattern load distribution expressions.
- The element can be used as a coupling between two parts (structural-solid) to transfer forces and moments. In comparison to MPCs and the kinematic coupling constraint, the distributing coupling element can be considered a more “flexible” connection.

Choosing an Appropriate Element

Two- and three-dimensional distributing coupling elements are available. Element DCOUP2D describes behavior only in the global *X*–*Y* plane. Element DCOUP2D can be used in an axisymmetric analysis; however, its use requires care in selecting the load distributing weight factors. For example, a uniform axial load distribution to a structure would require specification of load distribution weight factors in proportion to the radius of the coupling nodes. Since the radius of these nodes will change with deformation, this use of DCOUP2D would only approximate the correct load distribution behavior in a large-displacement analysis.

Defining the Distributing Coupling

To define a distributing coupling, you specify the coupling nodes to which loads are to be distributed, along with the corresponding weighting of the distribution. A minimum of two coupling nodes is required.

Input File Usage: **DISTRIBUTING COUPLING*, ELSET=*name*
node number or node set, weight_factor_1
node number or node set, weight_factor_2
...

Example

This example (see *Figure 1*) illustrates the use of the DCOUP3D element to impart a rotation to the surface of a structure that is expected to deform in a general way. In this case warping and motion within the plane of the end surface are expected to occur.

```
*ELEMENT, TYPE=DCOUP3D, ELSET=ROTATEELEMENT
1001, 1
*DISTRIBUTING COUPLING, ELSET=ROTATEELEMENT
COUPLESET, 1.0
…
*STEP, NLGEOM
…
*BOUNDARY
1, 6, 6, 1.0
…
*END STEP
```

Defining the Load Distribution

The element distributes loads such that the resultants of the forces on the coupling nodes are equal to the forces and moments on the element node. For cases of more than a few coupling nodes, the distribution of the forces is not determined by equilibrium alone, and the user-specified weight factors are used to define the distribution. The weight factors are dimensionless and are normalized within each element so that the sum of all weight factors is one. As a consequence, the normalized weight factors describe the proportion of the total element force and moment that is transmitted through the particular coupling node. In the case of transmission of forces alone, the

proportion of force transmitted through the node is simply the normalized weight factor. In the general case of transmission of forces and moments, the force distribution follows that of a classic bolt-pattern analysis, where the weight factors could be considered the areas of particular bolt cross-sections. Refer to *Distributing coupling constraints* for specific details of the load distribution.

In the example shown in *Figure 1* the weight factor distribution chosen is homogeneous, with a value of 1.0. For the rotation depicted, a more accurate load distribution would reflect the fact that the shear forces on nodes near the edge of the slot will diminish to zero, which could be described by choosing individual weight factors for nodes near the slot edge. If the loading on the element were along the axis of the structure, the homogeneous distribution shown would be appropriate. For cases where different loading modes require different descriptions of the weight factor distribution, multiple distributing coupling elements with different element nodes and different weight factors can be used.

Colinear Coupling Node Arrangements

The distributing coupling element transmits moments at the element node as a force distribution among the coupling nodes, even if these nodes have rotational degrees of freedom. Thus, when the coupling node arrangement is colinear, the element is not capable of transmitting all components of a moment at the element node. Specifically, the moment component that is parallel to the colinear coupling node arrangement will not be transmitted. When this case arises, a warning message is issued that identifies the axis about which the element will not transmit a moment.

Use with Nonuniform Meshes

When the distributing coupling element is used with coupling nodes attached to elements of varying size, care should be taken in selecting the weight factors. The weight factor selected for a node should generally scale with the size of the elements attached to that node.

Processing of Unattached Nodes

Cloud nodes that have no stiffness cause numerical singularities in Abaqus/Standard analyses. You can guide Abaqus/Standard to provide proper management of such nodes. By default, Abaqus/Standard issues an error message. You can direct Abaqus/Standard to remove or allow nodes that are not attached to any user elements. You should keep unattached nodes if they derive their stiffness by being main nodes to other nodes that have stiffness.

Input File Usage: Use the following option to direct Abaqus/Standard to issue an error message (default):

**DISTRIBUTING COUPLING*, FREE NODES=ERROR

Use the following option to direct Abaqus/Standard to remove nodes that are not attached to any user elements:

**DISTRIBUTING COUPLING*, FREE NODES=REMOVE

Use the following option to direct Abaqus/Standard to allow nodes that are not attached to any user elements:

**DISTRIBUTING COUPLING*, FREE NODES=ALLOW

Output

Element nodal forces (the force the element places on the element and coupling nodes) are available through element variable NFORC. Element kinetic energy is available in dynamic procedures through the whole element variable ELKE.

Distributing Coupling Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the distributing coupling elements available in Abaqus/Standard.

References:

- *Distributing Coupling Elements*
- **DISTRIBUTING COUPLING*

Element Types

DCOUP2D	Two-dimensional distributing coupling element
DCOUP3D	Three-dimensional distributing coupling element

Active Degrees of Freedom

DCOUP2D: 1, 2, 6

DCOUP3D: 1, 2, 3, 4, 5, 6

Additional Solution Variables

None.

Nodal Coordinates Required

DCOUP2D: *X*, *Y*

DCOUP3D: *X*, *Y*, *Z*

Element Property Definition

You must identify a minimum of two nodes to which the distributing coupling element distributes loads and mass; in addition, you can specify the element mass.

Input File Usage: **DISTRIBUTING COUPLING*

Element-Based Loading

None.

Element Output

ELKE	Element kinetic energy.
NFORC	Element nodal forces.

Nodes Associated with the Element

1 node is defined with the element. Additional nodes forming the coupling are defined in the element property definition.

Drag Chain Elements

In this section:

- *Drag Chains*
- *Drag Chain Element Library*

Drag Chains

Products: Abaqus/Standard

References:

- *Drag Chain Element Library*
- **DRAG CHAIN*
- **RIGID SURFACE*

Overview

Drag chain elements:

- are used for simulating the effects of drag chains on the seabed for near bottom bending simulation modeling; and
- can be used in two-dimensional or three-dimensional problems.

Typical Applications

The drag chain is modeled as a concentrated weight on the seabed, with a chain between it and an attachment point on the pipe (see *Figure 1*).

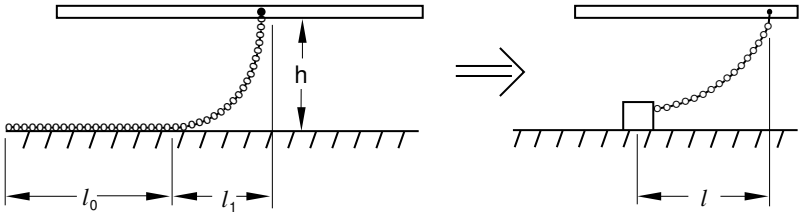


Figure 1: Drag chain model.

Given a uniform drag chain of total length ℓ_c, weight per unit length w, and friction coefficient μ between it and the seabed, attached to the pipeline at height h above the seabed, the length of chain on the seabed at slip, ℓ_0, is given by

$$\ell_0 = \ell_c \left[1 + \frac{\mu h}{\ell_c} - \left\{ \left(1 + \frac{\mu h}{\ell_c} \right)^2 - 1 + \left(\frac{h}{\ell_c} \right)^2 \right\}^{1/2} \right]$$

and the horizontal projection of the suspended length, ℓ_1, is

$$\ell_1 = \sqrt{2\mu h \ell_0}.$$

Thus, the equivalent model should have a friction limit of $\mu w \ell_0$. The horizontal length at slip, ℓ, can be taken as any value from ℓ_1 to $\ell_1 + \ell_0$. Comparison with experiment has shown that taking this length as $\ell = \ell_1 + \ell_0/2$ is a reasonable choice.

When the pipeline attachment point is directly above the weight, there will be no horizontal force or horizontal stiffness offered by a drag chain element; this position is assumed as the initial condition. As the pipe moves relative to the seabed, the horizontal force on the pipeline caused by the drag chain opposes the relative motion and gradually increases (an approximation to the catenary equation is used to relate the force to the offset ℓ) until

the drag chain slips when the force reaches the friction limit. The height, h, is assumed to be small compared to $\mu\ell_0$.

Choosing an Appropriate Element

Two- and three-dimensional drag chain elements are available.

Element DRAG2D assumes that the seabed is flat and parallel to the plane in which the pipe is moving; therefore, the seabed does not have to be modeled explicitly.

Element DRAG3D requires that the seabed be defined as an analytical rigid surface, which must be flat and parallel to the global (X, Y) plane and is considered to be fixed throughout the analysis.

Defining the Seabed for Three-Dimensional Drag Chains

The seabed is defined as an analytical rigid surface. This surface definition is used to determine if the chain touches the seabed, depending on the separation between the pipe node and the position of the seabed surface. See *Analytical Rigid Surface Definition* for more information.

Since the seabed is considered to be fixed, boundary conditions must be applied to the rigid body reference node of the seabed surface, which is also the second node of the DRAG3D element.

Input File Usage: Use the following option to define the seabed surface for DRAG3D elements:

**RIGID SURFACE*

In a model defined in terms of an assembly of part instances, the rigid surface definition that defines the seabed must appear inside the same part definition as the drag chain elements.

Defining the Drag Chain Behavior

For DRAG2D elements you specify the maximum horizontal length, ℓ, between the attachment point and the concentrated weight. At this length the weight will start to slip on the seabed. In addition, you specify the horizontal force between the weight and the seabed at slip (that is, the frictional limit).

For DRAG3D elements you specify the total length of the chain, the friction coefficient, and the weight per unit length of chain.

You must associate the drag chain behavior with a set of drag chain elements.

Input File Usage: **DRAG CHAIN*, ELSET=*name*
drag chain data

Drag Chain Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the drag chain elements available in Abaqus/Standard.

References:

- *Drag Chains*
- **DRAG CHAIN*
- **RIGID SURFACE*

Element Types

DRAG2D	Two-dimensional drag chain, for use in cases where only horizontal motion is being studied
DRAG3D	Three-dimensional drag chain

Active Degrees of Freedom

DRAG2D: 1, 2

DRAG3D: At the first node: 1, 2, 3. At the second node: 1, 2, 3, 4, 5, 6.

Additional Solution Variables

None.

Nodal Coordinates Required

DRAG2D: (X, Y) coordinates of the pipeline attachment node in the horizontal plane.

DRAG3D: (X, Y, Z) coordinates of both nodes.

Element Property Definition

Input File Usage: Use the following option to define the horizontal length at slip and the friction limit:

**DRAG CHAIN*

Use the following option to define the seabed for DRAG3D elements:

**RIGID SURFACE*

The rigid surface must be flat and parallel to the global (X, Y) plane.

Element-Based Loading

None.

Element Output

S11 The horizontal component of force supported by the drag chain in the plane parallel to the seabed.

S12 The vertical component of force in the drag chain for DRAG3D elements.

E11 The horizontal length of the drag chain for DRAG2D elements. The length of chain on the seabed floor (not suspended) for DRAG3D elements.

E12 The orientation of the drag chain (angle from the global *X*-axis).

Nodes Associated with the Element

DRAG2D: One node at the position where the chain attaches to the pipe.

DRAG3D: Two nodes. The first node is the node where the chain attaches to the pipe; the second node is the "reference node" of the rigid body containing the rigid surface that defines the seabed.

Elastic-Plastic Joints

In this section:

- *Elastic-Plastic Joints*
- *Elastic-Plastic Joint Element Library*

Elastic-Plastic Joints

Products: Abaqus/Aqua

References:

- **EPJOINT*
- *Elastic-Plastic Joint Element Library*

Overview

JOINT2D and JOINT3D elements:

- are available for use only in Abaqus/Aqua used in conjunction with Abaqus/Standard (*Abaqus/Aqua Analysis*);
- can be used to model flexible joints between structural members or the interaction between spud cans and the ocean floor;
- are valid for small displacements and rotations; and
- can be purely elastic or elastic-plastic.

Elastic-Plastic Joint Elements

Abaqus/Standard provides JOINT2D and JOINT3D elements for modeling a joint between structural members or between a structural member and a fixed support. They can be used in an Abaqus/Aqua analysis to model the interaction between a "spud can" and the sea floor for jack-up foundation analysis in offshore applications.

The joint has two nodes. One of these nodes should be constrained fully (by using a boundary condition) if the joint is between a structural member and a fixed support.

Kinematics and Local Coordinate System

The deformation of the joint is characterized by joint "strains," which are relative displacements and rotations between the nodes of the joint. The joint must be associated with a user-defined local orientation system (see *Orientations*) that is defined by three orthonormal directions: $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$.

The joint, when strained by relative extension or rotation of the two nodes, responds by applying equal and opposite forces and/or moments to the nodes. These forces and moments, or joint "stresses," can be a linear (elastic) or nonlinear (elastic-plastic) function of the "strains," depending on the type of constitutive model used in the joint.

The stresses and strains are named as shown in *Figure 1*. Positive stress indicates tension; positive strain indicates extension.

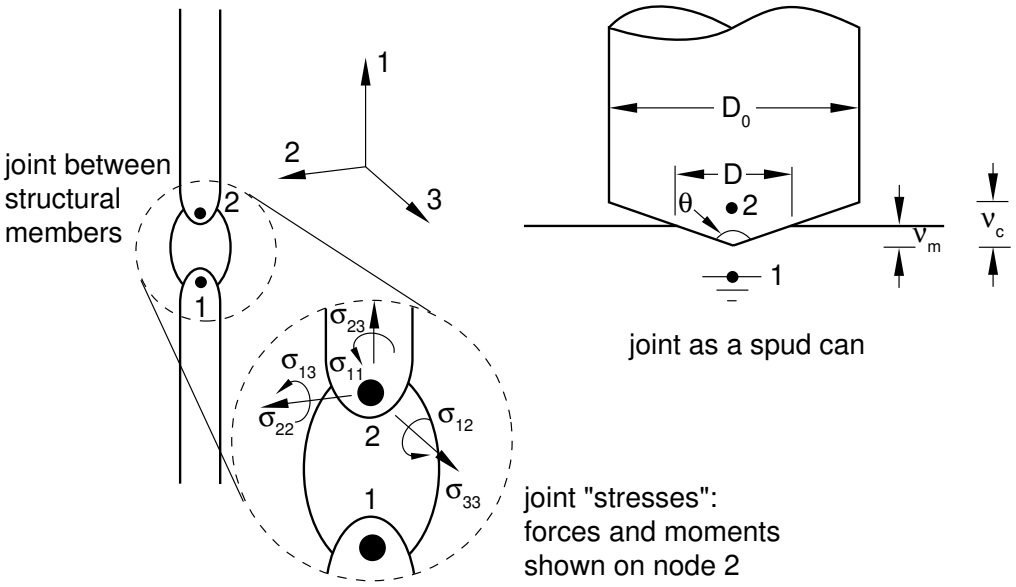


Figure 1: Local axis definition for joint elements.

Even when geometrically nonlinear analysis is requested (*Geometric Nonlinearity*), the element kinematics are defined with the assumption of small relative displacements and small rotations; therefore, these elements should not be used when these assumptions are violated. If large rotations are required and there is no plasticity, JOINTC elements can be used (see *Flexible Joint Element*).

The "extensional" strains are defined through

$$\varepsilon_{ii} = \Delta u \cdot e_i \qquad \text{(no summation)},$$

and the "bending" strains through

$$\varepsilon_{ij} = \Delta \phi \cdot \mathbf{e}_k, \qquad (\text{where} \quad i < j \quad \text{and} \quad k \neq i, j),$$

where

$$\Delta \mathbf{u} = \mathbf{u}^2 - \mathbf{u}^1 , \quad \Delta \phi = \phi^2 - \phi^1$$

are the relative displacements and rotations of the two nodes of the joint, respectively.

For two-dimensional elements only the axial strains ε_{11}, ε_{22}, and the bending strain ε_{12} exist. For three-dimensional elements all six components exist.

Input File Usage: Use the following option to associate a local orientation system with an elastic-plastic joint element:

**EPJOINT*, ORIENTATION=*name*

Joint Constitutive Models

The elastic moduli for joint elasticity can be entered in one of two ways. You can specify a general, anisotropic relation between the forces/moments and elastic extensions. Alternatively, you can enter moduli specific for a spud can; the elastic stiffness matrix is diagonal and depends on the diameter of the spud can at the soil surface, *D*, which can vary if spud can plasticity is defined and the spud can is conical. See *Joint Elasticity Models* below for details.

Three joint plasticity models are provided. Two are specific to spud cans. The third is a parabolic model for structural joints or members. See *Joint Plasticity* below for details.

If plasticity is included, the plastic straining is assumed to occur in the local 1–2 plane so that the only nonzero plastic strains are ε_{11}^{pl}, ε_{22}^{pl}, and ε_{12}^{pl}. It is assumed that plasticity in the 3-direction can be neglected. In a three-dimensional model strains out of the 1–2 plane produce purely elastic response.

If the parabolic plasticity model for structural joints or members is used, the 1-direction is the axial direction along the members, while the 2-direction is the transverse direction (see *Figure 1*). In the spud can plasticity models the 1-direction is the vertical direction, and the 2-direction is the horizontal direction in which plastic extension can take place. In three-dimensional models the 3-direction is the horizontal direction in which only elastic extension can take place.

Any combination of elastic and plastic models can be used. For example, usually spud can elastic moduli will be used with spud can plasticity, but the use of general moduli with spud can plasticity is allowed.

If plasticity is used in a three-dimensional model, coupling is not allowed through the elastic modulus between the strains or stresses in the 1–2 plane ($\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{12}$) and the remaining, out-of-plane, strains ($\varepsilon_{33}, \varepsilon_{13}, \varepsilon_{23}$). Thus, in this case many of the general elastic moduli must be set to zero.

Input File Usage: Use one or both of the following options immediately after the related **EPJOINT* option to define the joint constitutive model:

**JOINT ELASTICITY*
**JOINT PLASTICITY*

Orientation

Care must be taken in defining the local directions and node numbering so that the motion of node 2 relative to node 1 in the positive 1-direction of the local axis corresponds to extension. Incorrect specification of the local directions or element node numbering can produce incorrect results in plastic analysis because compression will be interpreted as extension.

If one of the nodes must be fixed to represent the ground, it is most convenient to let this node be the first node of the element; extension is then represented by the motion of node 2 of the element in the positive local 1-direction. If a spud can is being modeled in this way, the local 1-direction should be the outward normal to the ocean floor. For a two-dimensional analysis that uses Abaqus/Aqua structural loads, this direction must be the global *y*-direction.

For a three-dimensional analysis that uses Abaqus/Aqua structural loads, the local 1-direction should point in the global *z*-direction. If plasticity is being used, the local 2-direction should be set so that the 1–2 plane is the plane of greatest deformation.

Input File Usage: Use the following orientation definition to model a spud can with the first node fixed:

**ORIENTATION*, NAME=*name*, TYPE=RECTANGULAR
0, 1, 0, −1, 0, 0

Use the following orientation definition for a three-dimensional Abaqus/Aqua analysis with plasticity:

**ORIENTATION*, NAME=*name*, TYPE=RECTANGULAR
0, 0, 1, *x*, *y*, 0

where (*x*, *y*, 0) defines the local 2-direction.

Spud Can Geometry

If either spud can elasticity or spud can plasticity is used, you must specify the constants to define the spud can geometry. The entire spud can section definition has no effect if there is neither spud can elasticity nor spud can plasticity.

The spud can, illustrated in *Figure 1*, can be either conical-based or flat-based. The spud can geometry is defined by D_o, the diameter of the cylindrical portion, and θ, the planar angle of the conical portion, where $0 < \theta \leq 180°$. You can specify a flat-based spud can by omitting the specification of θ or by giving a value of 0 or 180 for θ.

Input File Usage: **EPJOINT*, SECTION=SPUD CAN
D_o, θ

Spud Can Initial Embedment

If spud can plasticity is defined or if there is spud can elasticity and the spud can is conical, you must specify the initial embedment of the spud can, ν_i.

The embedment can be prescribed directly or by specifying a "preload" that produces the embedment, as discussed below. Specification of both embedment and preload is not allowed. If either embedment or preload is given, both embedment and equivalent preload (in the case of plasticity) can be examined in the data file at the start of the analysis.

At any time in the analysis the spud can has a total (plastic) embedment of $\nu_m = \nu_i - \varepsilon_{11}^{pl}(t)$, where $\varepsilon_{11}^{pl}(t)$ is the plastic embedment between the start of the analysis and time t. (The negative sign in this equation reflects the fact that the sign convention for strain in Abaqus is positive for tensile strain. Most often for spud can plasticity, $\varepsilon_{11}^{pl}(t)$ will be compressive, or negative.) The joint can be purely elastic, in which case $\varepsilon_{11}^{pl} = 0$, so $\nu_m = \nu_i$ always.

The height of the conical portion of the spud can is given by $\nu_c = D_o/2\tan(\theta/2)$. The effective diameter of the spud can at the soil surface, D, is defined by

1. For a flat-based spud can:
$$D = D_o.$$
2. For a conical-based spud can:
   a. Cone portion partially penetrating ($\nu_m < \nu_c$):
   $$D = 2\nu_m \tan\frac{\theta}{2}$$
   b. Penetration beyond cone-cylinder transition ($\nu_m \geq \nu_c$):
   $$D = D_o.$$

The current spud can area at the soil surface, A, is defined through $A = \pi D^2/4$. The effective diameter can vary throughout the analysis only for a conical spud can with plasticity.

The embedment has no effect and is not required if the spud can is cylindrical and spud can plasticity is not defined.

Specifying the Embedment Directly

The embedment value can be prescribed directly using initial conditions (see *Initial Conditions*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=SPUD EMBEDMENT

Specifying the Spud Can Preload

If spud can plasticity is defined, you can specify the initial compressive capacity ("preload"), $V_c^{(i)}$, instead of the embedment. In this case Abaqus/Aqua will use the hardening law to calculate the plastic embedment that follows when the preload is applied vertically.

The preload initial condition is used only to calculate the initial plastic embedment; the spud can starts the analysis in a zero strain and stress state at this initial plastic embedment, and the preload is assumed to be removed. You must apply any operational vertical load through loading within the history definition.

Input File Usage: **INITIAL CONDITIONS*, TYPE=SPUD PRELOAD

Embedment in an Elastic Spud Can Analysis

If the spud can model is purely elastic, the spud can geometry is needed only for calculating the embedded diameter of the spud can for spud can elastic moduli. The embedment is required for this calculation only if the spud can is conical.

Output

Force and moment output in the element local system is available through the "stress" output variable S. Extension and relative rotation are available through the "strain" output variable E. Elastic and plastic strains are available through the output variables EE and PE. For spud cans the plastic embedment since the start of the analysis is available through the vertical component of plastic strain, PE11, and will usually be negative, indicating compression; the total vertical embedment, ν_m, is available through output variable PEEQ. Element nodal force (the force the element places on its nodes, in the global system) is available through element variable NFORC.

Joint Elasticity Models

The elastic load-displacement behavior of the JOINT2D and JOINT3D elements is characterized by elastic spring stiffnesses, which are assembled to form the elastic element stiffness matrix. A special diagonal modulus for spud cans can be specified or, alternatively, a fully populated (general) elastic modulus can be specified.

Spud Can Moduli

Spud can moduli can be prescribed for either two-dimensional or three-dimensional elements.

Two-Dimensional Spud Can Moduli

The elastic stiffness for a two-dimensional spud can is

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{Bmatrix} = \begin{bmatrix} k_{1111} & 0 & 0 \\ 0 & k_{2222} & 0 \\ 0 & 0 & k_{1212} \end{bmatrix} \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} \end{Bmatrix},$$

where

k_{1111}

is the vertical elastic spring stiffness, $2DG_{vv}/(1-\nu)$;

k_{2222}

is the horizontal elastic spring stiffness, $16\left(1-\nu\right)DG_{hh}/\left(7-8\nu\right)$;

k_{1212}

is the elastic spring stiffness in bending, $D^3G_{rr}/3\left(1-\nu\right)$;

in which G_{vv}, G_{hh}, and G_{rr} are equivalent elastic shear moduli for vertical, horizontal, and rotational displacements, respectively; ν is the Poisson's ratio of the soil (suggested value: 0.2 for sand and 0.5 for clay).

Input File Usage: **JOINT ELASTICITY*, MODULI=SPUD CAN, NDIM=2

Three-Dimensional Spud Can Moduli

For a three-dimensional spud can the moduli are

$$\begin{Bmatrix}\sigma_{11}\\ \sigma_{22}\\ \sigma_{33}\\ \sigma_{12}\\ \sigma_{13}\\ \sigma_{23}\end{Bmatrix}=\begin{bmatrix}k_{1111} & 0 & 0 & 0 & 0 & 0\\ 0 & k_{2222} & 0 & 0 & 0 & 0\\ 0 & 0 & k_{3333} & 0 & 0 & 0\\ 0 & 0 & 0 & k_{1212} & 0 & 0\\ 0 & 0 & 0 & 0 & k_{1313} & 0\\ 0 & 0 & 0 & 0 & 0 & k_{2323}\end{bmatrix}\begin{Bmatrix}\varepsilon_{11}\\ \varepsilon_{22}\\ \varepsilon_{33}\\ \varepsilon_{12}\\ \varepsilon_{13}\\ \varepsilon_{23}\end{Bmatrix},$$

where

k_{1111}

is the vertical elastic spring stiffness, $2DG_{vv}/\left(1-\nu\right)$;

k_{2222}

is a horizontal elastic spring stiffness, $16\left(1-\nu\right)DG_{hh}/\left(7-8\nu\right)$;

k_{3333}

is a horizontal elastic spring stiffness, $16\left(1-\nu\right)DG_{hh}/\left(7-8\nu\right)$;

k_{1212}

is an elastic spring stiffness in bending, $D^3G_{rr}/3\left(1-\nu\right)$;

k_{1313}

is an elastic spring stiffness in bending, $D^3G_{rr}/3\left(1-\nu\right)$;

k_{2323}

is the torsional elastic spring stiffness, k_t;

in which G_{vv}, G_{hh}, G_{rr}, and ν are as before and k_t is a user-specified torsional stiffness value.

Straining out of the 1–2 plane through the strains $\varepsilon_{33}, \varepsilon_{13}$, and ε_{23} produces purely elastic response in the three-dimensional model regardless of plasticity. The moduli related to these strains are assumed not to be affected by the plasticity so that k_{3333}, k_{1313}, and k_{2323} are based on the initial embedded diameter, while the other moduli depend on the current embedded diameter.

Input File Usage: **JOINT ELASTICITY*, MODULI=SPUD CAN, NDIM=3

General Moduli

General moduli can be specified for either two-dimensional or three-dimensional elements.

Two-Dimensional General Moduli

For the two-dimensional case six independent elastic moduli are needed. The stress-strain relations are as follows:

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{12} \end{Bmatrix} = \begin{bmatrix} k_{1111} & k_{1122} & k_{1112} \\ & k_{2222} & k_{2212} \\ sym & & k_{1212} \end{bmatrix} \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{12} \end{Bmatrix}$$

Input File Usage: **JOINT ELASTICITY*, MODULI=GENERAL, NDIM=2

Three-Dimensional General Moduli

For the three-dimensional case 21 independent elastic moduli are needed. The stress-strain relations are as follows:

$$\begin{Bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \sigma_{12} \\ \sigma_{13} \\ \sigma_{23} \end{Bmatrix} = \begin{bmatrix} k_{1111} & k_{1122} & k_{1133} & k_{1112} & k_{1113} & k_{1123} \\ & k_{2222} & k_{2233} & k_{2212} & k_{2213} & k_{2223} \\ & & k_{3333} & k_{3312} & k_{3313} & k_{3323} \\ & & & k_{1212} & k_{1213} & k_{1223} \\ & sym & & & k_{1313} & k_{1323} \\ & & & & & k_{2323} \end{bmatrix} \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{12} \\ \varepsilon_{13} \\ \varepsilon_{23} \end{Bmatrix} = \left[D^{el}\right] \begin{Bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \varepsilon_{12} \\ \varepsilon_{13} \\ \varepsilon_{23} \end{Bmatrix}.$$

Input File Usage: **JOINT ELASTICITY*, MODULI=GENERAL, NDIM=3

Joint Plasticity

In what follows $V = -\sigma_{11}, H = \sigma_{22}$, and $M = \sigma_{12}$ represent the vertical *compressive* load, the horizontal load in the 1–2 plane, and the bending moment in the local 1–2 plane, respectively.

If plasticity is defined, the joint can yield axially, horizontally, or rotationally. The stress depends linearly on the elastic strain. The elastic moduli can depend on the plasticity in the case of a conical spud can, through the diameter at the surface, *D*.

The models are rate independent, with a yield equation of the form

$$f(\boldsymbol{\sigma}, \mathbf{H}) \leq 0,$$

where f is the yield function and $\mathbf{H}$ is a set of hardening parameters, which in these models depend on total vertical plastic embedment, ν_m; the form of f and the definition of $\mathbf{H}$ defines the type of plasticity model.

The flow rule requires that the plastic flow direction is normal to the contours of the flow potential, g. Associated flow is assumed in all of these models (except at vertices in the yield surface, as discussed below).

Yield Surface

The three available plasticity models all use parabolic yield surfaces. Each has a compressive and a tensile limit for the stress in the 1-direction, which are termed V_c and V_t, respectively; V_t is zero for the clay model. The sign convention for V_c and V_t is such that they are always positive; thus, $V = -\sigma_{11}$ always obeys

$$-V_t \leq V \leq V_c.$$

The yield surface is most conveniently drawn in $(\overline{V}, \overline{R})$-space, where $\overline{V}$ is normalized compressive vertical load and is defined as

$$\overline{V} = \frac{V - V_o}{V_u},$$

where $V_o = \frac{1}{2}(V_c - V_t)$ is the middle value of the limiting elastic range for V, and $V_u = \frac{1}{2}(V_c + V_t)$ is the length of the limiting range for V. The normalized load is, therefore, always within the range

$$-1 \leq \overline{V} \leq 1,$$

with $\overline{V} = -1$ representing the tensile limit $V = -V_t$ and $\overline{V} = 1$ representing the compressive limit $V = V_c$. $\overline{R}$ is the normalized equivalent horizontal load and is defined through

$$\overline{R} = \sqrt{\left(\frac{M}{M_m}\right)^2 + \left(\frac{H}{H_m}\right)^2},$$

where M_m and H_m are the moment and horizontal yield stresses. The normalized moment and normalized horizontal force are defined through $\overline{M} = M/M_m$ and $\overline{H} = H/H_m$.

The normalized yield function in $(\overline{V}, \overline{R})$-space for each model is defined through

$$f = \overline{R} + \overline{V}^2 - 1$$

and is a parabola as plotted in *Figure 2*. The yield surface in the space of the three normalized stresses $(\overline{V}, \overline{M}, \overline{H})$ is the surface of revolution of this parabola.

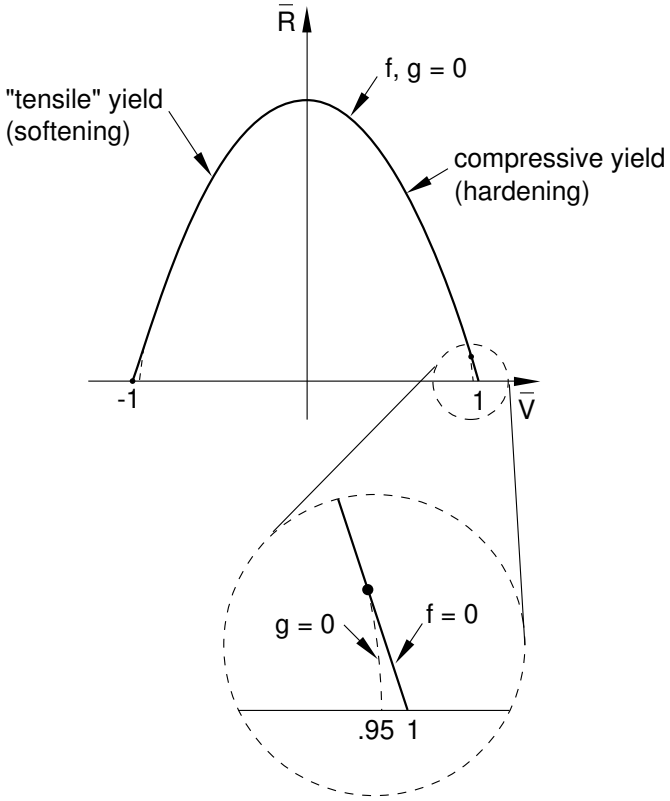


Figure 2: Yield surface and flow potential contour.

Flow Potential

The flow potential is the same as the yield function (associated flow) except that some smoothing is done to the flow potential where the yield function has corners.

The yield surface has corners and, therefore, nonunique normals at points where it is intersected by the $\overline{V}$-axis.

To avoid problems with the indeterminate flow directions at these corners, Abaqus/Standard uses a flow potential whose contours are rounded in the region of the vertex, as indicated in the detail of a vertex shown in *Figure 2*.

This rounding is achieved by fitting an elliptical segment to the flow potential contour for $|V| \geq 0.95$.

Integration of the Plasticity Equations

Abaqus/Aqua uses fully implicit integration for the plasticity equations. The corresponding tangent stiffness is unsymmetric for these plasticity models. By default, the symmetrized tangent is used in the global Newton loop. If the convergence rate seems to be poor, you may get some benefit out of using the unsymmetric matrix storage and solution scheme for the step (see *Defining an Analysis*).

Joint Plasticity Models

The three models differ only in the definitions of V_c, V_t, M_m, and H_m and in the hardening definitions. We present the yield function for each model as it is presented in the literature rather than in normalized form. The equivalent normalized form can be obtained by identifying M_m and H_m, which are explicit in the given yield functions for clay and member plasticity; for the sand model they are provided for reference.

Sand Model

1. Yield function:

$$f = \sqrt{\left(\frac{M}{DV_c}\right)^2 + \Lambda_1\left(\frac{H}{V_c}\right)^2} + \Lambda_2\left[\left(\frac{V}{V_c}\right)^2 - \left(1 - \frac{V_t}{V_c}\right)\frac{V}{V_c} - \frac{V_t}{V_c}\right] = 0,$$

   where Λ_1 and Λ_2 are constant coefficients that determine the geometric shape of the yield function. The special case of $\Lambda_1 = 1.0, \Lambda_2 = 0.5,$ and $V_t = 0.0$ gives the yield function as proposed by Osborne, et al.

2. Work hardening equations:

   a. Flat-base spud can:

$$\frac{V_c}{AD_o\gamma} = 0.3\ N_\gamma\left(1 - e^{-\alpha\nu_m/D_o}\right) + N_q\nu_m/D_o,$$

      where γ is soil unit weight; α is an experimentally determined constant; and N_γ and N_q are classical bearing capacity factors, which can be calculated as:

$$N_q = e^{\pi\tan\phi}\tan^2\left(45 + \frac{\phi}{2}\right),$$
$$N_\gamma = 2\left(N_q + 1\right)\tan\phi,$$

      where ϕ is the soil friction angle.

   b. Conical-base spud can:

      a. Cone portion partially penetrating:

$$\frac{V_c}{AD\gamma} = 0.3N_\gamma\left(1 - e^{-\alpha\beta\nu_m/D}\right) + N_q\beta\nu_m/D.$$

      b. Penetration beyond cone-cylinder transition:

$$\frac{V_c}{AD_o\gamma} = 0.3N_\gamma\left(1 - e^{-\alpha(\nu_m - \nu_c + \beta\nu_c)/D_o}\right) + N_q\left(\nu_m - \nu_c + \beta\nu_c\right)/D_o,$$

         where β is a "cone equivalency coefficient."

The constants α and β are based on the following empirical relation, which has been derived from centrifuge data:

$$\alpha = 1.954 \times 10^{-9}\phi^{6.129}$$
$$\beta = 0.71 - 0.014\phi$$

in which the soil friction angle ϕ is in degrees.

The sand model yield function can be put in normalized form by using $M_m = \kappa DV_c$ and $H_m = \kappa V_c\Lambda_1^{-.5}$, where $\kappa = \Lambda_2(1 + V_t/V_c)^2/4$. For the model of Osborne et al. $\kappa = 1/8$.

This model requires a nonzero initial embedment or equivalent preload.

Input File Usage: **JOINT PLASTICITY*, TYPE=SAND

Clay Model

1. Yield function:

$$f = \sqrt{\left(\frac{M}{8M_m}\right)^2 + \left(\frac{H}{8H_m}\right)^2} - 0.5\left(\frac{V}{V_c}\right)\left(1 - \frac{V}{V_c}\right) = 0,$$

where

$$\begin{aligned} M_m &= \frac{V_c D}{3\pi}, \\ H_m &= s_u\left(A + 2A_h\right). \end{aligned}$$

s_u is the undrained shear strength of clay; and A_h is the elevation area of the embedded portion of the spud can, defined through:

   a. Flat-base spud can:

$$A_h = D_o \nu_m$$

   b. Conical-base spud can:

      a. Cone portion penetrating:

$$A_h = 0.5 D \nu_m = \nu_m^2 \tan\frac{\theta}{2}$$

      b. Penetration beyond cone-cylinder transition:

$$A_h = D_o\left(\nu_m - \frac{0.25 D_o}{\tan\frac{\theta}{2}}\right)$$

2. Work hardening equations:

   a. Flat-base spud can:

$$V_c = a + b\nu_m$$

   b. Conical-base spud can:

$$V_c = \frac{\nu_m - c}{a + b\nu_m}$$

   where a, b, and c are user-defined empirical constants.

This model has zero yield strength in tension $(V_t = 0)$ and requires a nonzero initial embedment or equivalent preload.

Input File Usage: **JOINT PLASTICITY*, TYPE=CLAY

Parabolic Model for Structural Joints/Members

1. Yield function:

$$f = \sqrt{\left(\frac{M}{M_m}\right)^2 + \left(\frac{H}{H_m}\right)^2} + \left(\frac{V - V_o}{V_u}\right)^2 - 1 = 0,$$

where M_m, H_m are horizontal and moment capacities, respectively.

2. Work hardening: no work hardening is assumed (the model is perfectly plastic).

Input File Usage: **JOINT PLASTICITY*, TYPE=MEMBER

Plasticity Analysis Issues

Because associated flow is assumed in the spud can plasticity models, tensile vertical plastic strain can occur whenever the yield surface is encountered with $\overline{V} < 0$. It is not required that the vertical force itself be tensile for tensile plastic yield to occur; tensile plastic yield can occur on any part of the yield surface where $V < V_o$. The spud can models soften during this tensile plastic yield; if there is insufficient support from the rest of the model, an instability can occur and the analysis may fail to converge. When this happens, the spud can is likely to be lifting out of the sea floor.

To make it easier to diagnose analysis problems that may arise due to these issues, a message is printed to the message file in the following cases: if tensile plastic yield occurs for a spud can, if yield occurs near the top of the parabolic yield surface ($\overline{V} < 0.1$) where there is very little hardening, or if the embedment of a spud can becomes less than 10% of the initial embedment. These messages are not printed more than once in a given step.

The plasticity algorithm can fail in an iteration if the strain increment is excessively large. Some details that may be of help in diagnosing failure in joint elements can be obtained by requesting detailed printout to the message file of problems with the plasticity algorithms (see *The Abaqus/Standard Message File*).

Elastic-Plastic Joint Element Library

Products: Abaqus/Aqua

Overview

This section provides a reference to the elastic-plastic joint elements available in Abaqus/Aqua.

References:

- *Elastic-Plastic Joints*
- **EPJOINT*

Element Types

JOINT2D	Two-dimensional elastic-plastic joint element
JOINT3D	Three-dimensional elastic-plastic joint element

Active Degrees of Freedom

1, 2, 6 for JOINT2D

1, 2, 3, 4, 5, 6 for JOINT3D

Additional Solution Variables

None.

Nodal Coordinates Required

None.

Element Property Definition

Input File Usage: **EPJOINT*

Element-Based Loading

None.

Element Output

The relative displacements and rotations corresponding to the forces and moments below are chosen by requesting the corresponding "strains." Elastic and plastic strains are available. For a spud can the vertical (plastic) embedment since the start of the analysis is given by PE11; the total vertical embedment is available as PEEQ.

JOINT2D

S11 Total direct force in the first local direction.

S22 Total direct force in the second local direction.

S12 Total moment about the third local direction.

JOINT3D

S11 Total direct force in the first local direction.

S22 Total direct force in the second local direction.

S33 Total direct force in the third local direction.

S12 Total moment about the third local direction.

S13 Total moment about the second local direction.

S23 Total moment about the first local direction.

Nodes Associated with the Element

Two nodes.

Eulerian Elements

In this section:

- *Eulerian Elements*
- *Eulerian Element Library*

Eulerian Elements

Products: Abaqus/Explicit Abaqus/CAE

References:

- *Eulerian Analysis*
- *Eulerian Element Library*
- **EULERIAN SECTION*
- *Creating Eulerian sections*

Overview

Eulerian elements:

- can be used only in explicit dynamic analyses;
- must have eight unique nodes;
- are filled with void material by default;
- can be initialized with nonvoid material;
- can contain multiple materials simultaneously; and
- can be partially filled with material.

Typical Applications

Eulerian elements are useful for simulations involving material that undergoes extreme deformation, up to and including fluid flow. The Eulerian formulation allows material to flow from one element to another, even as the Eulerian mesh remains fixed. Applications that utilize Eulerian elements are discussed in *Eulerian analysis of a collapsing water column* and *Rivet forming*.

For more information on Eulerian analyses, see *Eulerian Analysis*.

Choosing an Appropriate Element

The available Eulerian elements are the three-dimensional, 8-node element EC3D8R and the three-dimensional, 8-node thermally coupled element EC3D8RT. Two-dimensional simulations can be approximated using a one-element thick mesh or a wedge-shaped mesh with appropriate boundary conditions. The Eulerian mesh is typically a simple rectangular grid of elements that does not conform to the shape of the Eulerian materials. Complex material shapes can be represented inside this mesh using a combination of fully and partially filled elements surrounded by void regions.

Defining the Eulerian Element's Section Properties

You must associate the Eulerian section definition with a set of Eulerian elements. This set of elements must not share nodes with other types of elements. The section definition provides a list of materials that may occupy the Eulerian elements.

Input File Usage: **EULERIAN SECTION*, ELSET=*element_set_name*
data lines giving list of materials

Abaqus/CAE Usage: Property module: **Create Section**: select **Solid** as the section **Category** and **Eulerian** as the section **Type**
Assign->Section: select part

Eulerian Element Library

Products: Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the Eulerian elements available in Abaqus/Explicit.

References:

- *Eulerian Analysis*
- **EULERIAN SECTION*

Element Types

Eulerian Stress/Displacement Element

EC3D8R 8-node linear brick, multimaterial, reduced integration with hourglass control

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

None.

Eulerian Thermally Coupled Element

EC3D8RT 8-node thermally coupled linear brick, multimaterial, reduced integration with hourglass control

Active Degrees of Freedom

1, 2, 3,11

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

You must specify a list of materials that might be present in the Eulerian element. You can also assign a material instance name to each material (see *Eulerian Section Definition*).

Input File Usage: **EULERIAN SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Solid** as the section **Category** and **Eulerian** as the section **Type**

Element-Based Loading

Distributed Loads

Distributed loads are available only for Eulerian elements. They are specified as described in *Distributed Loads*.

Load ID (*DLOAD): BX
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *X*-direction.

Load ID (*DLOAD): BY
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *Y*-direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Body force in global *Z*-direction.

Load ID (*DLOAD): BXNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *X*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BYNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *Y*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-3}
Description: Nonuniform body force in global *Z*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity

Units: LT^{-2}

Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): P*n*
Abaqus/CAE Load/Interaction: **Pressure**
Units: FL^{-2}

Description: Pressure on face *n*.

Load ID (*DLOAD): P*n*NU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Nonuniform pressure on face *n* with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): SBF
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-5}T^2$

Description: Stagnation body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): SP*n*
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T^2$

Description: Stagnation pressure on face *n*.

Load ID (*DLOAD): TRSHR*n*
Abaqus/CAE Load/Interaction: **Surface traction**
Units: FL^{-2}

Description: Shear traction on face *n*.

Load ID (*DLOAD): TRVEC*n*
Abaqus/CAE Load/Interaction: **Surface traction**
Units: FL^{-2}

Description: General traction on face *n*.

Load ID (*DLOAD): VBF
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T$

Description: Viscous body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): VP*n*
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-3}T$

Description: Viscous pressure on face *n*, applying a pressure proportional to the velocity normal to the face and opposing the motion.

Distributed Heat Fluxes

Distributed heat fluxes are available only for EC3D8RT elements. They are specified as described in *Thermal Loads*.

Load ID (*DFLUX): BF
Abaqus/CAE Load/Interaction: **Body heat flux**
Units: $JL^{-3}T^{-1}$

Description: Heat body flux per unit volume.

Load ID (*DFLUX): S*n*
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Heat surface flux per unit area into face *n*.

Film Conditions

Film conditions are available only for EC3D8RT elements. They are specified as described in *Thermal Loads*.

Load ID (*FILM): F*n*
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\theta^{-1}$
Description: Film coefficient and sink temperature (units of θ) provided on face *n*.

Radiation Types

Radiation conditions are available only for EC3D8RT elements. They are specified as described in *Thermal Loads*.

Load ID (*RADIATE): R*n*
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless
Description: Emissivity and sink temperature (units of θ) provided on face *n*.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are available for Eulerian elements. They are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element surface.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Nonuniform pressure on the element surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DSLOAD): SP
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-4}T^{2}$
Description: Stagnation pressure on the element surface.

Load ID (*DSLOAD): ST
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-1}T$

Description: Surface tension on the Eulerian material surface.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element surface.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element surface.

Load ID (*DSLOAD): VP
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-3}T$
Description: Viscous pressure applied on the element surface. The viscous pressure is proportional to the velocity normal to the element face and opposing the motion.

Distributed Heat Fluxes

Surface-based heat fluxes are available only for EC3D8RT elements. They are specified as described in *Thermal Loads*.

Load ID (*DSFLUX): S
Abaqus/CAE Load/Interaction: Surface heat flux
Units: $JL^{-2}T^{-1}$
Description: Heat surface flux per unit area into the element surface.

Film Conditions

Surface-based film conditions are available only for EC3D8RT elements. They are specified as described in *Thermal Loads*.

Load ID (*SFILM): F
Abaqus/CAE Load/Interaction: Surface film condition
Units: $JL^{-2}T^{-1}\theta^{-1}$
Description: Film coefficient and sink temperature (units of θ) provided on the element surface.

Radiation Types

Surface-based radiation conditions are available only for EC3D8RT elements. They are specified as described in *Thermal Loads*.

Load ID (*SRADIATE): R
Abaqus/CAE Load/Interaction: Surface radiation
Units: Dimensionless
Description: Emissivity and sink temperature (units of θ) provided on the element surface.

Element Output

A set of output variables is written for each Eulerian material instance listed in the Eulerian section definition. The output variable names are automatically appended with the material instance names. For example, if you

define material instances named “steel” and “tin” and request stress output, the first stress components will be written to separate output variables named “S11_steel” and “S11_tin.”

All output is given in global coordinates.

Stress and Other Tensor Components

Stress and other tensors (excluding total strain tensors) are available. All tensors have the same components. For example, the stress components are as follows:

S11 ***XX***, direct stress.

S22 ***YY***, direct stress.

S33 ***ZZ***, direct stress.

S12 ***XY***, shear stress.

S13 ***XZ***, shear stress.

S23 ***YZ***, shear stress.

Element-Averaged Quantities

Several output variables are also available as element-averaged quantities. These variables are computed as a volume fraction weighted average of all materials present in the element. Use of these variables can substantially decrease the size of the output database for models with many Eulerian materials. For example:

SVAVG Volume fraction averaged stress.

Node Ordering and Face Numbering on Elements

All elements must have eight nodes. Degenerate elements are not supported.

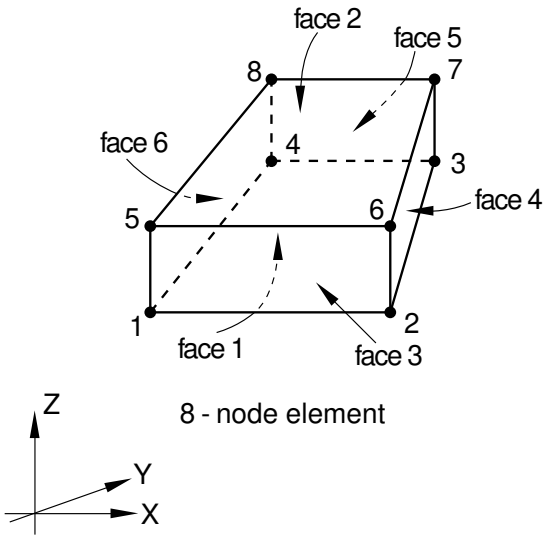


Table 1: Element faces

Face 1	1 – 2 – 3 – 4 face
Face 2	5 – 8 – 7 – 6 face
Face 3	1 – 5 – 6 – 2 face
Face 4	2 – 6 – 7 – 3 face
Face 5	3 – 7 – 8 – 4 face
Face 6	4 – 8 – 5 – 1 face

Numbering of Integration Points for Output

The single integration point is located at the centroid of the element. All materials within the element are evaluated at this integration point.

Flexible Joint Elements

In this section:

- *Flexible Joint Element*
- *Flexible Joint Element Library*

Flexible Joint Element

Products: Abaqus/Standard

References:

- *Flexible Joint Element Library*
- **JOINT*
- **DASHPOT*
- **SPRING*

Overview

JOINTC elements:

- are used to model joint interactions; and
- are made up of translational and rotational springs and parallel dashpots in a local, corotational coordinate system.

Details of the element formulation can be found in *Flexible joint element*.

Typical Applications

The JOINTC element is provided to model the interaction between two nodes that are (almost) coincident geometrically and that represent a joint with internal stiffness and/or damping (such as a rubber bushing in a car suspension system) so that the second node of the joint can displace and rotate slightly with respect to the first node.

Joints that have only one or two axes of rotation and no relative displacement are better modeled by the REVOLUTE- or UNIVERSAL-type MPCs (see *General Multi-Point Constraints*).

Similar functionality is available using connectors; see *About Connectors*.

Defining the Joint Behavior

The joint behavior consists of linear or nonlinear springs and dashpots in parallel, coupling the corresponding components of relative displacement and of relative rotation in the joint. You define the spring and dashpot behavior as described in *Springs* and *Dashpots*.

Each spring or dashpot definition defines the behavior for one of the six local directions; up to six spring and six dashpot definitions can be included. If no specification is given for a particular local relative motion in the joint, the joint is assumed to have no stiffness with respect to that component.

The joint behavior can be defined in a local coordinate system that rotates with the motion of the first node of the element (*Orientations*). If a local coordinate system is not defined, the global system is used.

You must associate the joint behavior with a set of JOINTC elements.

The kinematic behavior of JOINTC elements is described in detail in *Flexible joint element*.

Input File Usage: Use the following options to define the joint behavior:

**JOINT*, ELSET=*name*, ORIENTATION=*name*
**DASHPOT*
**SPRING*
Up to six **SPRING* and **DASHPOT* options can appear.

Using JOINTC Elements in Large-Displacement Analyses

In large-displacement analysis the formulation for the relationship between moments and rotations limits the usefulness of these elements to small relative rotations. The relative rotation across a JOINTC element should be of a magnitude to qualify as a small rotation.

Flexible Joint Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the flexible joint elements available in Abaqus/Standard.

References:

- *Flexible Joint Element*
- **JOINT*

Element Types

JOINTC Joint interaction element

Active Degrees of Freedom

1, 2, 3, 4, 5, 6

Additional Solution Variables

None.

Nodal Coordinates Required

None. The element nodes do not need to have coordinates defined since the action associated with these elements is defined by specifying the degrees of freedom involved.

Element Property Definition

Input File Usage: **JOINT*

Element-Based Loading

None.

Element Output

S11 Total direct force in the first local direction.

S22 Total direct force in the second local direction.

S33 Total direct force in the third local direction.

S12 Total moment about the first local direction.

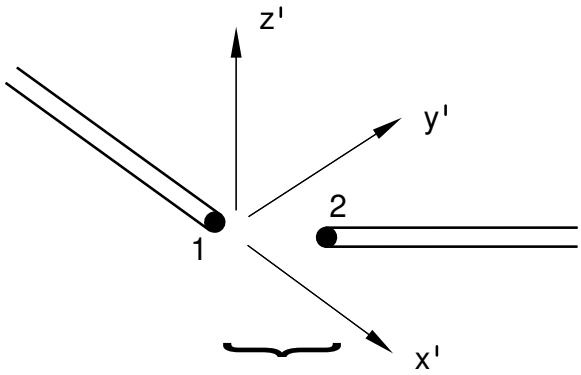
S13 Total moment about the second local direction.

S23 Total moment about the third local direction.

The relative displacements and rotations corresponding to the forces and moments above are chosen by requesting the corresponding “strains.”

Nodes Associated with the Element

Two nodes. The rotation at the first node of the element defines the rotation of the local axis system.



(local system, defined by a local orientation, attached to node 1)

Fluid Pipe Connector Elements

Fluid pipe connector elements suitable for modeling the junction between two pipes are available in Abaqus/Standard.

In this section:

- *Fluid Pipe Connector Elements*
- *Thermal Fluid Pipe Connector Elements*
- *Slurry Fluid Pipe Connector Elements*
- *Fluid Pipe Connector Element Library*

Fluid Pipe Connector Elements

Products: Abaqus/Standard

References:

- *Fluid Pipe Connector Element Library*
- **FLUID PIPE CONNECTOR SECTION*
- **FLUID PIPE CONNECTOR LOSS*

Overview

Fluid pipe connector elements:

- allow you to simulate discrete viscous pressure loss terms in a fluid pipe network; and
- can be used to simulate control valves allowing you to reduce or increase the resistance to flow or alternatively to turn the flow off.

Fluid pipe connectors in Abaqus/Standard use a pure pressure formulation to model steady-state flow of a single-phase, incompressible fluid through a fully filled junction in a pipe network.

Typical Applications

Fluid pipe connector elements are typically used to simulate the junction between two or more fluid pipe elements (see *Fluid Pipe Elements*) such as a valve, a T-connector, or a diffuser.

Choosing an Appropriate Element

Several fluid pipe connector element types are available. For two-dimensional and axisymmetric analyses, use element type FPC2D2. For three-dimensional analyses, use element type FPC3D2.

Assigning a Material Definition to a Set of Fluid Pipe Connector Elements

You must associate a material definition with each connector element section property.

The material that is defined for the fluid pipe connector section refers to the fluid that is flowing through the connector. The properties that must be defined for the fluid are the pore fluid density and viscosity. For the viscosity definition fluid pipe connector elements support both Newtonian and non-Newtonian fluids. The supported non-Newtonian fluids are power law, Bingham Plastic, and Herschel-Bulkley models (see *Viscosity*).

Input File Usage: Use the following options to specify the pore fluid density and viscosity:

**FLUID PIPE CONNECTOR SECTION*, MATERIAL=*material name*
**MATERIAL*, NAME=*material name*
**DENSITY*, PORE FLUID
**VISCOSITY*, DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC

Multiple Fluid Flow

Abaqus allows the use of multiple fluids in an analysis, up to a maximum of four fluid types. This capability is useful in situations where the spatial distribution of the different fluid types is known (that is, precomputed) at

all times during the analysis. In other words, the type of fluid at each node in the domain is known at all times during the analysis.

The definition of multiple fluids in an analysis involves associating each fluid with a unique integer and a specific predefined field variable. This integer acts as a fluid identifier. You can utilize this fluid identifier to specify the properties (for example, viscosity) of each fluid. The field variable associated with a fluid type allows you to predefine the fluid type at a node as a function of time using amplitude definitions or as functions of both time and space utilizing user subroutine *UFIELD* or *USDFLD*.

You specify a value for each field variable that is associated with a fluid identifier at each node. It is recommended that you specify a value for the field variable (FV) such that $0.0 \leq FV \leq 1.0$. Abaqus assigns the fluid type at an integration point to be the one for which the field variable has the maximum value at that point. If all field variables have the same numerical value at an integration point, the first fluid (fluid identifier 1) is assumed to be active.

Input File Usage: **VISCOSITY*, FLUID ID=*Fluid Id number*,DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC
**FLUID ID FIELD*
Fluid Id number, *field_id*

Fluid Pipe Connector Equations

The geometry of a fluid pipe connector element is expressed in terms of hydraulic area and hydraulic diameter. The hydraulic diameter is expressed in terms of the cross-sectional area, A, and the wetted perimeter, P, as

$$D_h = \frac{(4A)}{P}.$$

A fluid pipe connector element is defined by two nodes. Unlike the fluid pipe elements, the geometric length of fluid pipe connector elements plays no role in the fluid equilibrium equations and, therefore, the fluid pipe connector nodes should be modeled as being coincident or having the same elevation. If the nodes are not coincident, you might introduce an artificial flow in the connector element due to hydrostatic pressure imbalance. The viscous pressure loss across a fluid pipe connector in Abaqus/Standard is given as

$$\triangle P = K\frac{\rho V^2}{2},$$

where

- P_1, P_2 are the pressures at the nodes and $\triangle P = (P_1 - P_2)$;
- V is the fluid velocity in the pipe;
- ρ is the fluid density; and
- K is a loss term.

The mass flow rate Q through the connector can be related to the fluid and pipe parameters as $Q = \rho AV$.

Specifying the Fluid Pipe Connector Geometry and Connector Loss

Abaqus/Standard supports four different types of fluid pipe connector loss terms:

- Standard connection type with bidirectional loss terms;
- A Hooper2K connector;
- A Darby3K connector; and
- A user subroutine that you can use to define bidirectional loss terms.

Specifying Standard Connector Loss Terms

The standard fluid pipe connector uses constant bidirectional loss terms K_1 and K_2 that you define. If the flow is from local node 1 to node 2, the total pressure loss is

$$\triangle P = K_1 \frac{\rho V^2}{2};$$

and if the flow is from local node 2 to node 1, the dynamic pressure loss is

$$\triangle P = K_2 \frac{\rho V^2}{2}.$$

Input File Usage: **FLUID PIPE CONNECTOR LOSS*, TYPE=CONNECTION

Specifying the Connector Loss Based on Reynold's Number

This method utilizes the Hooper 2K parameters or Darby 3K parameters. The K values for different types of connectors and valves can be found in the literature. The 2K parameter or 3K parameter methods are sometimes preferable to constant bidirectional loss terms because they include a Reynold's number dependence. (The Hooper 2K parameter has a unit dependence that requires the hydraulic diameter to be defined in inches.) Irrespective of the flow direction, a flow-dependent loss value is computed during the analysis and is given by

$$\triangle P = K \frac{\rho V^2}{2}.$$

The Hooper 2K loss term is defined as

$$K = \frac{K_1}{Re} + K_\infty \left(1 + \frac{1}{D_h}\right),$$

where K_1 and K_∞ are constant loss terms. The Darby 3K loss term is defined as

$$K = \frac{K_1}{Re} + K_\infty \left(1 + \frac{K_d}{D_h^{0.3}}\right),$$

where K_1, K_∞, and K_d are constant loss terms.

Input File Usage: **FLUID PIPE CONNECTOR LOSS*, TYPE=HOOPER2K
**FLUID PIPE CONNECTOR LOSS*, TYPE=DARBY3K

Specifying the Connector Loss with a User Subroutine

You can specify bidirectional connector loss terms (K_1 and K_2) for fluid pipe connector elements using user subroutine *UFLUIDCONNECTORLOSS*. As with the standard connector, if the flow is from local node 1 to node 2, the total pressure loss is

$$\triangle P = K_1 \frac{\rho V^2}{2};$$

and if the flow is from local node 2 to node 1, the dynamic pressure loss is

$$\triangle P = K_2 \frac{\rho V^2}{2}.$$

Input File Usage: **FLUID PIPE CONNECTOR LOSS*, TYPE=USER

Specifying the Laminar Flow Transition for Low Reynolds Number Flows

You can specify the laminar flow transition parameter that is used to switch flow computations from a purely laminar, linear formulation to a nonlinear iterative formulation. The Hooper 2K and Darby 3K methods include Reynold's number dependence. Therefore, the laminar flow transition can be used only when the connector loss is defined by either one of these types. This ensures better convergence when the flow in the connector is zero

or close to zero in magnitude. The default laminar transition flow Reynold's number is 2500. User subroutine *UFLUIDCONNECTORLOSS* is not called when the computed *Re* is less than the default or specified value.

Input File Usage: **FLUID PIPE CONNECTOR LOSS*,
LAMINAR FLOW TRANSITION=*Reynold's number value*

Specifying the Control Valve Behavior

You can control the flow in the connector by simulating the presence of a control valve. By default, no valve behavior is defined and the fluid is fully flowing. When activated, user subroutine *UFLUIDCONNECTORVALVE* is called to determine the valve opening whose value must be between 0.0 and 1.0. The valve control option is valid only with the Hooper 2K and Darby 3K connector loss methods. This is because the flow in the connector can be set to zero, and the use of laminar flow transition gives better convergence behavior under these conditions.

Input File Usage: **FLUID PIPE CONNECTOR LOSS*, VALVE CONTROL=USER

Specifying Initial and Prescribed Conditions

You can define a field distribution over the nodes of the fluid pipe connector elements.

Input File Usage: Use one or both of the following options:

**INITIAL CONDITIONS*, TYPE=FIELD

Specifying Loads and Boundary Conditions

Fluid pipe connector elements allow for the specification of pressure boundary conditions and volumetric flow rates at the nodes. The flow rate must be a nonzero value. At a particular node, either a pressure or flow rate can be specified but not both. Since the fluid pipe connectors do not use the geometric length in the fluid equilibrium equations, gravity loads are not supported for these elements.

Input File Usage: Use the following option to specify the pressure at the inlet or outlet:

**BOUNDARY*
node or node set, *8*, *8*, *magnitude*

Use the following option to specify the flow rate at the inlet or outlet:

**CFLOW*
node or node set, , *magnitude*

Thermal Fluid Pipe Connector Elements

Products: Abaqus/Standard

References:

- *Fluid Pipe Connector Element Library*
- **FLUID PIPE CONNECTOR SECTION*
- **FLUID PIPE CONNECTOR LOSS*
- **FLUID PIPE CONNECTOR THERMAL LOSS*
- **FLUID PIPE CONNECTOR WALL*

Overview

Thermal fluid pipe connector elements allow you to simulate discrete viscous pressure loss and discrete thermal loss terms in a fluid pipe network.
You can use thermal fluid pipe connector elements to simulate control valves, allowing you to reduce or increase the resistance to flow or, alternatively, to turn off the flow.

Thermal fluid pipe connectors in Abaqus/Standard use a pure pressure formulation to model steady-state flow of a single-phase, incompressible fluid through a fully filled junction in a pipe network.

Typical Applications

Thermal fluid pipe connector elements are typically used to simulate the junction between two or more thermal fluid pipe elements (see *Thermal Fluid Pipe Elements*) such as a valve, a T-connector, or a diffuser.

Choosing an Appropriate Element

Several thermal fluid pipe connector elements are available. They can be broadly classified as *regular* thermal fluid pipe connector elements or *annular* thermal fluid pipe connector elements. Element types FPC2D2T and FPC3D2T belong to the class of regular thermal fluid pipe connector elements. Element types FPC2D2AT and FPC3D2AT belong to the class of annular thermal fluid pipe connector elements that enable the modeling of the annulus region of a connector. For a detailed discussion of the differences between the two classes (*regular* versus *annular*) of thermal fluid pipe elements, see *Choosing an Appropriate Element* in *Thermal Fluid Pipe Elements*. Similar considerations carry over for the corresponding connector elements.

The geometric layout of the regular and annular thermal fluid pipe connector elements and the corresponding element connectivities are shown in the figures below. The nodes at each end are geometrically coincident. Even though the figures show these elements to have finite lengths, the geometric length of the connectors (based on the nodal coordinates of its two ends) is ignored in the pressure drop computations.

You can use an offset to create the pipe wall nodes of both the regular and the annular thermal fluid pipe connector elements.

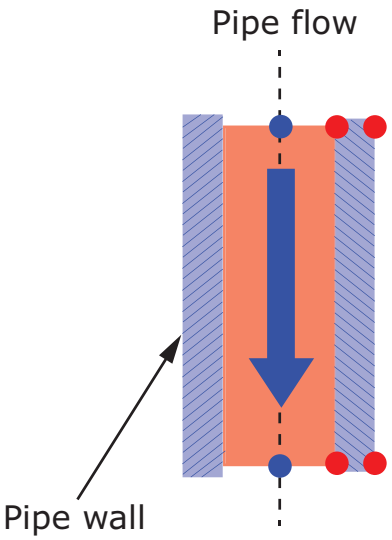


Figure 1: Modeling pipe flow.

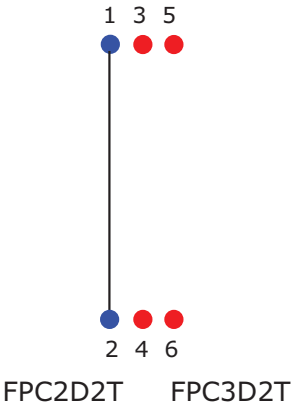


Figure 2: Element connectivity for thermal fluid pipe connector elements.

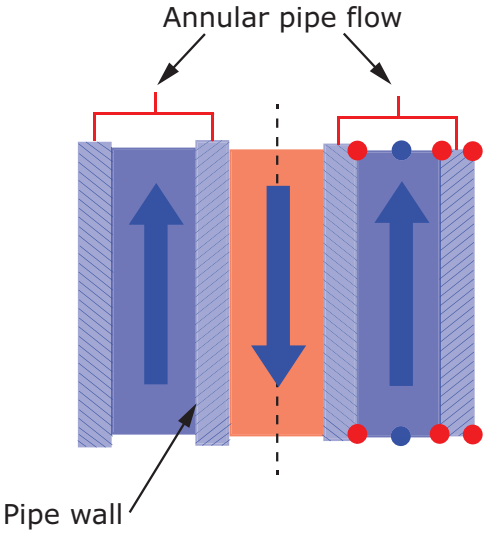


Figure 3: Modeling annular pipe flow.

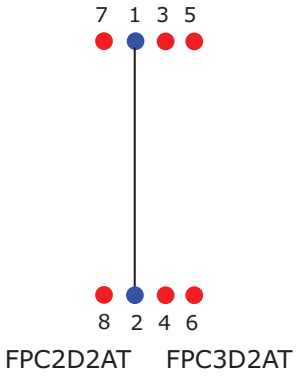


Figure 4: Element connectivity for annular thermal fluid pipe connector elements.

When you define nodes for the thermal fluid pipe connectors and annular thermal fluid pipe connector elements, the nodes wind up sharing the same geometric location; therefore, it is difficult to distinguish the nodes that form the inner wall of the connector from those for the outer wall or the center fluid region. To make selection easier, Abaqus/Standard defines multiple surfaces on a thermal pipe connector element, as shown in *Fluid Pipe Connector Element Library*. These surface definitions allow you to easily select the different nodes on the thermal fluid pipe connector elements to be used to specify a coupling to the formation or to another pipe or connector element.

Input File Usage: Use the following option to define an offset used to create the connector pipe wall nodes:

**ELEMENT*, OFFSET=*number*

Assigning a Material Definition to a Set of Thermal Fluid Pipe Connector Elements

You must associate a material definition with each connector element section property.

The material that is defined for the thermal fluid pipe connector section refers to the fluid that is flowing through the connector. You must define the following properties for the fluid: fluid density and viscosity. For the viscosity definition, thermal fluid pipe connector elements support both Newtonian and non-Newtonian fluids. The following non-Newtonian fluid models are supported: power law, Bingham Plastic, and Herschel-Bulkley models (see *Viscosity*).

For the thermal fluid pipe connector elements, you must also specify the following thermal properties:

- Conductivity and specific heat for the fluid
- Density, conductivity, and specific heat for the pipe wall

Input File Usage: Use the following options to specify the flow and thermal properties of the fluid:

**FLUID PIPE CONNECTOR SECTION*, MATERIAL=*material name*
**MATERIAL*, NAME=*material name*
**VISCOSITY*, DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC
**DENSITY*, PORE FLUID
**CONDUCTIVITY*, PORE FLUID
**SPECIFIC HEAT*, PORE FLUID

In addition, use the following options to specify the thermal properties for the pipe wall connector:

**DENSITY*
**CONDUCTIVITY*
**SPECIFIC HEAT*

Multiple Fluid Flow

Abaqus allows the use of multiple fluids in an analysis, up to a maximum of four fluid types. This capability is useful in situations where the spatial distribution of the different fluid types is known (that is, precomputed) at all times during the analysis. In other words, the type of fluid at each node in the domain is known at all times during the analysis.

The definition of multiple fluids in an analysis involves associating each fluid with a unique integer and a specific predefined field variable. This integer acts as a fluid identifier. You can utilize this fluid identifier to specify the properties (for example, viscosity) of each fluid. The field variable associated with a fluid type allows you to predefine the fluid type at a node as a function of time using amplitude definitions, or as functions of both time and space utilizing user subroutine *UFIELD* or *USDFLD*.

You specify a value for each field variable, which is associated with a fluid identifier, at each node. It is recommended that you specify a value for the field variable (FV) such that, $0.0 \leq FV \leq 1.0$. Abaqus assigns the fluid type at an integration point to be the one for which the field variable has the maximum value at that point. If all field variables have the same numerical value at an integration point, the first fluid (fluid identifer 1) is assumed to be active.

Input File Usage: **VISCOSITY*, FLUID ID=*Fluid Id number*, DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC
**FLUID ID FIELD*
Fluid Id number, *field_id*

Thermal Fluid Pipe Connector Equations

The main governing equations for the fluid flow and pressure loss can be found in *Fluid Pipe Connector Equations*. The different options of defining discrete pressure loss can be found in *Specifying the Fluid Pipe Connector Geometry and Connector Loss*. You can also define a control valve behavior in thermal fluid pipe connector elements as described in *Specifying the Control Valve Behavior*.

Specifying the Wall Diameters for Thermal Fluid Pipe Connector Elements

You must define the geometry of the connector wall as a circular cross-section by specifying the inner and outer wall radii.

Input File Usage: **FLUID PIPE CONNECTOR WALL*

Specifying the Thermal Loss in the Fluid

You can define the thermal loss due to the fluid flow in the thermal fluid pipe connector elements.

Input File Usage: **FLUID PIPE CONNECTOR THERMAL LOSS*

Specifying Initial and Prescribed Conditions

You can define an initial temperature or field distribution over the nodes of the thermal fluid pipe connector elements.

Input File Usage: Use one or both of the following options:

**INITIAL CONDITIONS*, TYPE=TEMPERATURE
**INITIAL CONDITIONS*, TYPE=FIELD

Specifying Loads and Boundary Conditions

Thermal fluid pipe connector elements allow for the specification of pressure boundary conditions and volumetric flow rates at the nodes. The flow rate must be a nonzero value. At a particular node, you can specify either a pressure or flow rate but not both. Because the thermal fluid pipe connector elements do not use the geometric length in the fluid equilibrium equations, gravity loads are not supported for these elements. You can specify boundary conditions on the temperature degree of freedom.

Input File Usage: Use the following option to specify the pressure at the inlet or outlet:

**BOUNDARY*
node or node set, *8*, *8*, *magnitude*

Use the following option to specify the flow rate at the inlet or outlet:

**CFLOW*
node or node set, , *magnitude*

Use the following option to specify the temperature at the inlet or outlet:

**BOUNDARY*
node or node set, *11*, *11*, *magnitude*

Slurry Fluid Pipe Connector Elements

Products: Abaqus/Standard

References:

- *Fluid Pipe Connector Element Library*
- **FLUID PIPE CONNECTOR SECTION*
- **FLUID PIPE CONNECTOR LOSS*
- **FLUID PIPE CONNECTOR THERMAL LOSS*
- **FLUID PIPE CONNECTOR WALL*

Overview

Slurry fluid pipe connector elements allow you to simulate discrete viscous pressure loss due to slurry flow and discrete thermal loss terms in a fluid pipe network.
You can use slurry fluid pipe connector elements to simulate control valves, allowing you to reduce or increase the resistance to flow or, alternatively, to turn off the flow.

A slurry is a mixture of a carrier fluid and solid proppant particles. Slurry fluid pipe connectors in Abaqus/Standard use a pure pressure formulation to model steady-state flow of a single-phase, incompressible fluid through a fully filled junction in a pipe network. In addition to pressure, the nodes of these elements also have slurry concentration and, optionally, temperature degrees of freedom. The slurry concentration refers to the ratio of the volume of the solid proppant particles to that of the slurry.

Typical Applications

Slurry fluid pipe connector and thermal-slurry fluid pipe connector elements are typically used to simulate the junction between two or more slurry fluid pipe and thermal-slurry fluid pipe elements (see *Slurry Fluid Pipe Elements*) such as a valve, a T-connector, or a diffuser.

Choosing an Appropriate Element

Several connector elements with only slurry concentration and pore pressure or with slurry concentration, pore pressure, and thermal degrees of freedom are available. They can be broadly classified as follows:

- Regular slurry fluid pipe connector elements (FPC2D2C and FPC3D2C)
- Regular thermal-slurry fluid pipe connector elements (FPC2D2Q and FPC3D2Q)
- Annular thermal-slurry fluid pipe connector elements (FPC2D2AQ and FPC3D2AQ), which enable modeling of the annulus region of a connector

For a detailed discussion of the differences between the two classes (*regular* versus *annular*) of fluid pipe elements, see *Choosing an Appropriate Element*. Similar considerations apply to the corresponding slurry fluid pipe connector elements.

The geometric layout of the regular and annular thermal-slurry fluid pipe connector elements and the corresponding element connectivities are similar to that shown for regular and annular thermal fluid pipe connector elements in *Thermal Fluid Pipe Connector Elements*. For information about defining surfaces on regular and annular fluid pipe connector elements, see *Fluid Pipe Connector Element Library*.

Assigning a Material Definition to a Set of Thermal Fluid Pipe Connector Elements

You must associate a material definition with each connector element section property.

The material that is defined for the slurry fluid pipe connector section refers to the fluid that is flowing through the connector. You must define the fluid density and the viscosity of the slurry to describe its flow behavior. The density of the slurry can be specified to be a function of the slurry concentration. In this case, the density is computed using the density of the fluid and the solid proppant particles, as well as the slurry concentration, using a rule of mixtures. The viscosity of the slurry can be specified as a function of the slurry concentration directly in tabular form. For the viscosity definition, slurry fluid pipe elements support both Newtonian and non-Newtonian fluids. The following non-Newtonian fluid models are supported: power law, Bingham Plastic, and Herschel-Bulkley models (see *Viscosity*).

For the thermal slurry fluid pipe connector elements, you must also specify the following thermal properties:

- Conductivity and specific heat of the slurry. The conductivity can be specified, optionally, as a function of the slurry concentration (as well as temperature and field variables, if necessary) directly in tabular form. The specific heat can be computed based on the specific heats of the carrying fluid and the solid proppant particles, as well as the slurry concentration, using a rule of mixtures.
- Density, conductivity, and specific heat for the pipe wall.

Input File Usage: Use the following options to specify the flow and thermal properties of the fluid:

**FLUID PIPE CONNECTOR SECTION*, MATERIAL=*material name*
**MATERIAL*, NAME=*material name*
**VISCOSITY*, DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC, SLURRY
**DENSITY*, PORE FLUID, SLURRY
**CONDUCTIVITY*, PORE FLUID, SLURRY
**SPECIFIC HEAT*, PORE FLUID, SLURRY

In addition, use the following options to specify the thermal properties for the pipe wall connector:

**DENSITY*
**CONDUCTIVITY*
**SPECIFIC HEAT*

Multiple Fluid Flow

Abaqus allows the use of multiple fluids in an analysis, up to a maximum of four fluid types. This capability is useful in situations where the spatial distribution of the different fluid types is known (precomputed) at all times during the analysis. In other words, the type of fluid at each node in the domain is known at all times during the analysis.

The definition of multiple fluids in an analysis involves assigning a unique integer value that acts as an identifier for each fluid type, and associating each fluid identifier with a specific predefined field variable used to specify the spatial distribution of the fluid as function of time. You can use the fluid identifier to specify the properties (for example, viscosity) of each fluid. The field variable associated with a fluid type allows you to predefine the fluid type at a node as a function of time using amplitude definitions or as functions of both time and space utilizing user subroutine *UFIELD* or *USDFLD*.

You specify a value for each field variable, associated with a fluid identifier, at each node. It is recommended that you specify a value for the field variable (FV) such that $0.0 \leq FV \leq 1.0$. Abaqus assigns the fluid type at an integration point to be the one for which the field variable has the maximum value at that point. If all field variables have the same numerical value at an integration point, the first fluid (fluid identifier equal to 1) is assumed to be active.

Input File Usage: **VISCOSITY*, FLUID ID=*Fluid Id number*, DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC, SLURRY
**FLUID ID FIELD*
Fluid Id number, *field_id*

Slurry Fluid Pipe Connector Equations

The main governing equations for the fluid flow and pressure loss are described in *Fluid Pipe Connector Equations*. For information about the different options for defining discrete pressure loss, see *Specifying the Fluid Pipe Connector Geometry and Connector Loss*. You can also define control valve behavior in slurry fluid pipe connector elements as described in *Specifying the Control Valve Behavior*. The main governing equations for the thermal behavior are described in *Thermal Fluid Pipe Connector Equations*. For the thermal-slurry fluid pipe connector elements, the thermal loss and connector wall diameters can be specified as shown in *Specifying the Thermal Loss in the Fluid* and *Specifying the Wall Diameters for Thermal Fluid Pipe Connector Elements*.

When you specify the connector loss terms for the fluid pressure loss as described in *Specifying the Fluid Pipe Connector Geometry and Connector Loss*, you must ensure that these terms account for the flow of slurry (in other words, they should not define only the carrier fluid). The density that is used for computing the pressure loss is the mixture density (computed using the density of the fluid and the solid proppant particles) and the slurry concentration, using a rule of mixtures. The slurry concentration stays constant through the slurry fluid pipe connector element and thermal-slurry fluid pipe connector element.

Specifying Initial and Prescribed Conditions

You can define an initial slurry concentration, an initial temperature, and a field distribution over the nodes of the slurry fluid pipe connector element and thermal-slurry fluid pipe connector element.

Input File Usage: Use one or both of the following options:

**INITIAL CONDITIONS*, TYPE=TEMPERATURE
**INITIAL CONDITIONS*, TYPE=FIELD
**INITIAL CONDITIONS*, TYPE=SLURRYVF

Specifying Loads and Boundary Conditions

Slurry fluid pipe connector elements and thermal slurry fluid pipe connector elements allow for the specification of pressure boundary conditions and volumetric flow rates at the nodes. The flow rate must be a nonzero value. At a particular node, you can specify either a pressure or flow rate but not both. Because the slurry fluid pipe connector elements do not use the geometric length in the fluid equilibrium equations, gravity loads are not supported for these elements. You can also specify boundary conditions on the slurry concentration and temperature degree of freedom.

Input File Usage: Use the following option to specify the pressure at the inlet or outlet:

**BOUNDARY*
node or node set, *8*, *8*, *magnitude*

Use the following option to specify the flow rate at the inlet or outlet:

**CFLOW*
node or node set, , *magnitude*

Use the following option to specify the temperature at the inlet or outlet:

**BOUNDARY*
node or node set, *11*, *11*, *magnitude*

Use the following option to specify the slurry volume fraction at the inlet or outlet:

**BOUNDARY*
node or node set, *31*, *31*, *magnitude*

Fluid Pipe Connector Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the fluid pipe connector elements available in Abaqus/Standard.

References:

- *Fluid Pipe Connector Elements*
- *Thermal Fluid Pipe Connector Elements*
- **FLUID PIPE CONNECTOR SECTION*
- **FLUID PIPE CONNECTOR LOSS*
- **FLUID PIPE CONNECTOR THERMAL LOSS*
- **FLUID PIPE CONNECTOR WALL*

Element Types

2D Pore Pressure Element

FPC2D2	2-node linear

Active Degrees of Freedom

8

Additional Solution Variables

None.

2D Coupled Temperature—Pore Pressure Elements

FPC2D2T	6-node linear
FPC2D2AT	8-node linear, annular

Active Degrees of Freedom

8 at the first two nodes of element connectivity

11 at all nodes

Additional Solution Variables

None.

2D Coupled Slurry—Pore Pressure Element

FPC2D2C	2-node linear

Active Degrees of Freedom

8

31

Additional Solution Variables

None.

2D Coupled Temperature—Slurry—Pore Pressure Elements

FPC2D2Q	6-node linear
FPC2D2AQ	8-node linear, annular

Active Degrees of Freedom

8 at the first two nodes of element connectivity

11 at all nodes

31 at the first two nodes of element connectivity

Additional Solution Variables

None.

3D Pore Pressure Element

FPC3D2	2-node linear

Active Degrees of Freedom

8

Additional Solution Variables

None.

3D Coupled Temperature–Pore Pressure Elements

FPC3D2T	6-node linear
FPC3D2AT	8-node linear, annular

Active Degrees of Freedom

8 at the first two nodes of element connectivity

11 at all nodes

Additional Solution Variables

None.

3D Coupled Slurry—Pore Pressure Element

FPC3D2C	2-node linear

Active Degrees of Freedom

8

31

Additional Solution Variables

None.

3D Coupled Temperature—Slurry—Pore Pressure Elements

FPC3D2Q	6-node linear
FPC3D2AQ	8-node linear, annular

Active Degrees of Freedom

8 at the first two nodes of element connectivity

11 at all nodes

31 at the first two nodes of element connectivity

Additional Solution Variables

None.

Nodal Coordinates Required

2D: *X*, *Y*

3D: *X*, *Y*, *Z*

Element Property Definition

The fluid pipe connector elements FPC2D2, FPC2D2C, FPC3D2, and FPC3D2C are defined by two nodes that can be coincident. If the nodes are not coincident, the length of the fluid pipe connector element is ignored in the viscous loss computations.

The thermal fluid pipe connector elements FPC2D2T and FPC3D2T have six nodes. Two nodes define the fluid pressure and temperature, and four nodes define the wall temperatures.The nodes that define the pipe wall are coincident with the nodes that define the pore pressure degrees of freedom.

The thermal-slurry fluid pipe connector elements FPC2D2Q and FPC3D2Q have six nodes. Two nodes define the fluid pressure, temperature, and slurry; and four nodes define the wall temperatures. The nodes that define the pipe wall are coincident with the nodes that define the pore pressure degrees of freedom.

The annular thermal fluid pipe connector elements FPC2D2AT and FPC3D2AT have eight nodes. Two nodes define the fluid pressure and temperature, and the remaining six nodes define the wall temperatures.

The annular thermal-slurry fluid pipe connector elements FPC2D2AQ and FPC3D2AQ have eight nodes. Two nodes define the fluid pressure, slurry, and temperature; and the remaining six nodes define the wall temperatures.

You must associate a fluid pipe connector section with a set of fluid pipe connector elements.

Input File Usage: **FLUID PIPE CONNECTOR SECTION*, ELSET=*name*

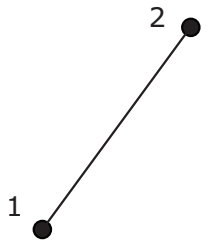
Element-Based Loading

None.

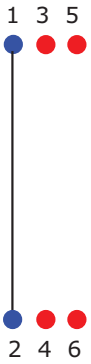
Element Output

FPDPRESS	Pressure drop across the element.

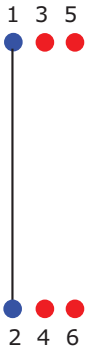
Node Ordering on Elements



FPC2D2 FPC3D2
FPC2D2C FPC3D2C



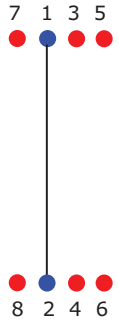
FPC2D2T FPC3D2T



FPC2D2Q FPC3D2Q



FPC2D2AT FPC3D2AT



FPC2D2AQ FPC3D2AQ

The following table displays the surface definition for coupled thermal-pore pressure fluid pipe elements (FPC2D2T, FPC3D2T) and coupled thermal-pore pressure slurry fluid pipe elements (FPC2D2Q, FPC3D2Q).

Face ID	Nodes
Face 1	5 – 6
Face 2	3 – 4
Face 3	1 – 2
Face 4	5
Face 5	3
Face 6	1
Face 7	6
Face 8	4
Face 9	2

The following table displays the surface definition for coupled thermal-pore pressure annulus fluid pipe elements (FPC2D2AT, FPC3D2AT) and coupled thermal-pore pressure slurry fluid pipe annulus elements (FPC2D2AQ, FPC3D2AQ).

Face ID	Nodes
Face 1	5 – 6
Face 2	3 – 4

Face ID	Nodes
Face 3	1 – 2
Face 4	5
Face 5	3
Face 6	1
Face 7	6
Face 8	4
Face 9	2
Face 10	7 – 8
Face 11	7
Face 12	8

Fluid Pipe Elements

Fluid pipe elements suitable for modeling incompressible pipe flow are available in Abaqus/Standard.

In this section:

- *Fluid Pipe Elements*
- *Thermal Fluid Pipe Elements*
- *Slurry Fluid Pipe Elements*
- *Fluid Pipe Element Library*

Fluid Pipe Elements

Products: Abaqus/Standard

References:

- *Fluid Pipe Element Library*
- **FLUID PIPE SECTION*
- **FLUID PIPE FLOW LOSS*

Overview

Fluid pipe elements in Abaqus/Standard allow you to simulate the viscous and gravity pressure loss terms in a fluid pipe network. The pipe elements use a pure pressure formulation and are based on Bernoulli's equation for the case of steady-state flow of a single-phase, incompressible fluid through a fully filled pipe with a constant cross-sectional area.

Typical Applications

Fluid pipe elements are used to simulate the flow of a liquid through a pipe or network of pipes to determine pressure drops and flow rates in a geostatic or coupled pore fluid diffusion/stress analysis (see *Geostatic Stress State* and *Coupled Pore Fluid Diffusion and Stress Analysis*). They can also be used to model one-dimensional wellbores in geomechanics.

Choosing an Appropriate Element

Several fluid pipe element types are available. For two-dimensional and axisymmetric analyses, use element type FP2D2. For three-dimensional analyses, use element type FP3D2.

Assigning a Material Definition to a Set of Fluid Pipe Elements

You must associate a material definition with each pipe element section property.

The material that is defined for the fluid pipe section refers to the fluid that is flowing through the pipe. The properties that must be defined for the fluid are the pore fluid density and viscosity. For the viscosity definition fluid pipe elements support both Newtonian and non-Newtonian fluids. The supported non-Newtonian fluids are power law, Bingham Plastic, and Herschel-Bulkley models (see *Viscosity*).

Input File Usage: Use the following options to specify the pore fluid density and viscosity:

**FLUID PIPE SECTION*, MATERIAL=*material name*
**MATERIAL*, NAME=*material name*
**DENSITY*, PORE FLUID
**VISCOSITY*, DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC

Multiple Fluid Flow

Abaqus allows the use of multiple fluids in an analysis, up to a maximum of four fluid types. This capability is useful in situations where the spatial distribution of the different fluid types is known (precomputed) at all times during the analysis. In other words, the type of fluid at each node in the domain is known at all times during the analysis.

The definition of multiple fluids in an analysis involves assigning a unique integer value that acts as an identifier for each fluid type, and associating each fluid identifier with a specific predefined field variable used to specify the spatial distribution of the fluid as function of time. You can utilize the fluid identifier to specify the viscosity of each fluid. Only the fluid viscosity can be different for different fluids. All other properties (that is, density, thermal) are assumed to be same for all the fluids. The field variable associated with a fluid type allows you to predefine the fluid type at a node as a function of time using amplitude definitions or as functions of both time and space utilizing user subroutine *UFIELD* or *USDFLD*.

You specify a value for each field variable, associated with a fluid identifier, at each node. It is recommended that you specify a value for the field variable (FV) such that $0.0 \leq FV \leq 1.0$. Abaqus assigns the fluid type at an integration point to be the one for which the field variable has the maximum value at that point. If all field variables have the same numerical value at an integration point, the first fluid (fluid identifier equal to 1) is assumed to be active.

Input File Usage: **VISCOSITY*, FLUID ID=*Fluid Id number*,DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC
**FIELD*, VARIABLE=*Field variable number*
**FLUID ID FIELD*
Fluid Id number,*Field variable number*

Using Fluid Pipe Elements in Symmetric Models

You can use fluid pipe elements in models that use symmetry to reduce the model size. The fluid pipe elements model circular geometry. You specify a symmetry value that is greater than 0 and less than or equal to 1. This value is interpreted as scaling of the circular pipe geometry in radians. A value of 1 corresponds to 2π radians, and a value of 0.5 corresponds to π radians. When you specify symmetry, you must scale the full model flow magnitude by the symmetry value. You do not need to apply any symmetric constraints or boundary conditions.

Input File Usage: **FLUID PIPE SECTION*, SYMMETRY=*value*

Fluid Pipe Equations

The geometry of a pipe element is expressed in terms of hydraulic area and hydraulic diameter. The hydraulic diameter is expressed in terms of the cross-sectional area (A) of the tube or channel and the wetted perimeter (P) as $D_h = \frac{(4A)}{P}$. A pipe element is defined by two noncoincident nodes. Using a Darcy-Weisbach approach, Bernoulli's equation (including viscous loss) between two points in space can be written as

$$\triangle P - \rho g \triangle Z = (C_L + K_i) \frac{\rho V^2}{2},$$

where

- P_1, P_2 are the pressures at the nodes and $\triangle P = (P_1 - P_2)$;
- Z_1, Z_2 are the elevations at the nodes and $\triangle Z = (Z_2 - Z_1)$;
- V is the fluid velocity in the pipe;
- ρ is the fluid density;
- g is the acceleration due to gravity;
- $C_L = \frac{fL}{D_h}$ is the loss coefficient;
- f is the friction factor of the pipe;
- L is the length of the pipe; and

- K_i is a directional loss term.

The assumption of constant cross-sectional area in a single element results in constant fluid velocity in a pipe element. The mass flow rate Q through the pipe can be related to the fluid and pipe parameters as $Q = \rho AV$.

Additional Loss Terms in Fluid Pipe Elements

The loss coefficient C_L can also include an added pipe length L_a as well as a pipe length scaling factor α. The general form of the loss coefficient is written as

$$C_L = \frac{f(L(1+\alpha)+L_a)}{D_h}.$$

In addition, you can also specify directional connection loss terms K_1 and K_2. If the flow is from local node 1 to node 2, the total pressure loss is

$$\triangle P - \rho g \triangle Z = (C_L + K_1)\frac{\rho V^2}{2};$$

and if the flow is from local node 2 to node 1, the dynamic pressure loss is

$$\triangle P - \rho g \triangle Z = (C_L + K_2)\frac{\rho V^2}{2}.$$

Specifying the Friction Loss Behavior

Abaqus/Standard supports four different methods for defining the friction factor (f):

- Blasius friction loss;
- Churchill friction loss;
- A tabular option; and
- A user subroutine.

Specifying Blasius Friction Loss Behavior for the Fluid Pipe Element

The Blasius friction loss method uses an empirical relation based on the Reynold's number (*Re*) to determine the friction factor. The method has two different regimes that depend on whether the flow is laminar or turbulent. There is a discontinuous jump in the friction factor when the flow transitions from laminar to turbulent at $Re = 2500$. The friction factor is empirically calculated as

$$f = \frac{64}{Re} : Re < 2500$$

$$f = \frac{0.3164}{Re^{0.25}} : Re \geq 2500.$$

Input File Usage: **FLUID PIPE FLOW LOSS*, TYPE=BLASIUS

Specifying Churchill Friction Loss Behavior for the Fluid Pipe Element

A more comprehensive formula that takes into account the pipe roughness K_s and captures the Moody's data accurately is the Churchill's formula. This formula transitions smoothly from laminar to turbulent flow. The friction factor is determined as

$$f = 8\left[\left(\frac{8}{Re}\right)^{12} + \frac{1}{(A+B)^{1.5}}\right]^{\frac{1}{12}},$$

$$A = \left[-2.457\ln\left(\left(\frac{7}{Re}\right)^{0.9} + 0.27\left(\frac{K_s}{D_h}\right)\right)\right]^{16},$$

$$B = \left(\frac{37350}{Re}\right)^{16}.$$

Input File Usage: **FLUID PIPE FLOW LOSS*, TYPE=CHURCHILL

Specifying the Friction Loss Behavior as a Table of Reynolds Number Versus Friction Factor

You can input a table of Re versus friction. Abaqus interpolates linearly between the values specified in the table. If one of the independent variables is outside the range of specified values, Abaqus uses the value that is closest in the table.

Input File Usage: **FLUID PIPE FLOW LOSS*, TYPE=TABULAR

Specifying the Friction Factor with a User Subroutine

You can specify the friction factor for the element with user subroutine *UFLUIDPIPEFRICTION*. The user subroutine is called by every fluid pipe element to determine the friction factor based on the fluid flow rates.

Input File Usage: **FLUID PIPE FLOW LOSS*, TYPE=USER

Specifying the Laminar Flow Transition for Low Reynolds Number Flows

You can specify the laminar flow transition parameter that is used to switch flow computations from a purely laminar, linear formulation to a nonlinear iterative formulation. The purely laminar formulation uses the Blasius friction factor when the computed Reynold's number is at or below the specified laminar flow transition number. This ensures better convergence when the flow in the pipe is zero or close to zero in magnitude. The default laminar transition flow Reynold's number is 2500. User subroutine *UFLUIDPIPEFRICTION* is not called when the computed Re is less than the default or specified value.

Input File Usage: **FLUID PIPE FLOW LOSS*,
LAMINAR FLOW TRANSITION=*Reynold's number value*

Specifying Initial and Prescribed Conditions

You can define a field distribution over the nodes of the fluid pipe elements.

Input File Usage: Use one or both of the following options:

**INITIAL CONDITIONS*, TYPE=FIELD

Specifying Loads and Boundary Conditions

Fluid pipe elements allow for the specification of pressure boundary conditions and volumetric flow rates at the nodes. At a particular node, either a pressure or flow rate can be specified but not both. You can also specify a gravity load on the fluid pipe element to determine the hydrostatic head at the nodes.

Input File Usage: Use the following option to specify the pressure at the inlet or outlet:

**BOUNDARY*
node or node set, *8*, *8*, *magnitude*

Use the following option to specify the flow rate at the inlet or outlet:

**CFLOW*
node or node set, , *magnitude*

Use the following option to specify a gravity load on the fluid pipe element:

**DLOAD*
element or element set, , *GRAV*, *gravity constant*, *comp1*, *comp2*, *comp3*

Thermal Fluid Pipe Elements

Products: Abaqus/Standard

References:

- *Fluid Pipe Element Library*
- **FLUID PIPE SECTION*
- **FLUID PIPE FLOW LOSS*
- **FLUID PIPE THERMAL*
- **FLUID PIPE WALL*

Overview

Thermal fluid pipe elements in Abaqus/Standard allow you to simulate the pressure loss due to viscous and gravity effects in a fluid pipe network, including the temperature field in the fluid flowing in the pipe, heat exchange between the fluid and the pipe, and heat transfer through the walls of the pipe.

The thermal fluid pipe elements model the flow field using a pure pressure formulation, based on Bernoulli's equation for steady-state flow of a single-phase, incompressible fluid through a fully filled pipe with a constant cross-sectional area. The thermal model includes convection and diffusion of heat in the fluid, as well as conduction of heat along the length and radial (thickness) directions in the pipe.

Typical Applications

Thermal fluid pipe elements are used to simulate:

- the flow of a liquid through a pipe or network of pipes to determine pressure drops and flow rates, and
- the temperature field in the flowing fluid and the pipe walls.

These elements can be used in a geostatic or coupled pore fluid diffusion/stress analysis (see *Geostatic Stress State* and *Coupled Pore Fluid Diffusion and Stress Analysis*). They can also be used to model one-dimensional wellbores in certain geotechnical applications.

Choosing an Appropriate Element

Several thermal fluid pipe elements are available. They can be broadly classified as follows:

- Regular thermal fluid pipe elements (FP2D2T and FP3D2T)
- Annular thermal fluid pipe elements (FP2D2AT and FP3D2AT)

For two-dimensional analyses, use element type FP2D2T or FP2D2AT. For three-dimensional analyses, use element type FP3D2T or FP3D2AT. A detailed discussion of the differences between the two classes (*regular* versus *annular*) follows.

Figure 1 shows a regular thermal fluid pipe element. It consists of a total of six nodes, three at each end of the element. Although each end has three distinct nodes, they actually share the same geometric location. The first node at each end (shown here in blue) is used to model the fluid flow and has both fluid pressure and temperature degrees of freedom. These nodes are referred to as the "flow nodes." The second and third nodes at each end (shown here in red) define the temperature variation through the wall of the pipe and have only the temperature degree of freedom. These nodes are referred to as the "wall nodes." *Figure 2* shows the element connectivity for the regular thermal fluid pipe elements.

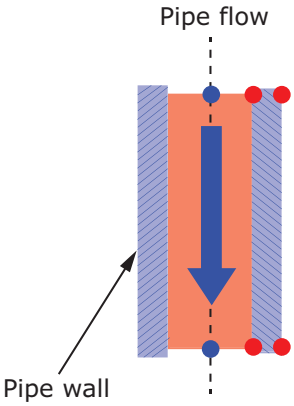


Figure 1: Modeling pipe flow.

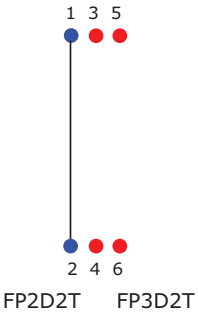


Figure 2: Element connectivity for thermal fluid pipe elements.

Figure 3 shows an annular pipe, which consists of a center region that is the same as a regular pipe (and should be modeled with a regular thermal fluid pipe element) and an annular region that surrounds the center region. The annular region (which must be modeled using an annular thermal fluid pipe element) has a total of eight nodes, four at each end of the pipe sharing the same geometric location. The nodes at the middle of the annulus (shown here in blue) have both fluid pressure and temperature degrees of freedom; they are referred to as the "flow nodes." The "wall nodes" (shown here in red) have only temperature degrees of freedom and model the temperature variation in the inner and outer walls of the annulus. *Figure 4* shows the element connectivity for the annular thermal fluid pipe element.

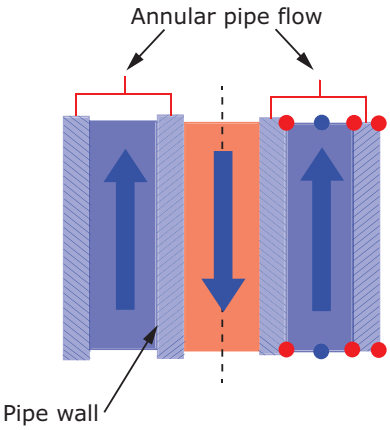


Figure 3: Modeling annular pipe flow.

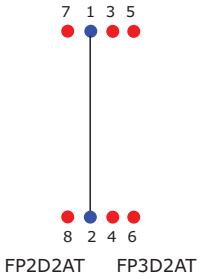


Figure 4: Element connectivity for annular thermal fluid pipe elements.

A typical application of the annular pipe element involves modeling the return flow in a piping system. As shown in *Figure 3*, the fluid flow directions in the center and the annular regions might be opposite to each other. The incoming fluid at the center region and the outgoing fluid in the annular region can be at different temperatures. The temperature degree of freedom at the outermost wall nodes of the regular pipe region (nodes 5 and 6 in *Figure 2*) and that at the corresponding innermost wall nodes of the annular region (nodes 7 and 8 in *Figure 4*) must be tied to ensure continuity of the temperature field between the center and the annular regions. In addition, if return flow is modeled, the fluid pressure and temperature degrees of freedom of the bottom "flow node" of the regular thermal fluid pipe element (node 2 in *Figure 2*) must be tied to the corresponding degrees of freedom of the bottom "flow node" of the annular thermal fluid pipe element (node 2 in *Figure 4*).

You can use an offset to create the pipe wall nodes of the regular thermal fluid pipe and annular thermal fluid pipe elements.

You can render thermal pipe elements and annular thermal pipe elements using the Visualization module of Abaqus/CAE. For more information, see *Controlling the display of thermal fluid pipe elements*.

Input File Usage: Use the following option to define an offset used to create the pipe wall nodes:

**ELEMENT*, OFFSET=*number*

When you define nodes for the thermal fluid pipe and annular thermal fluid pipe elements, the nodes wind up sharing the same geometric location; therefore, it is difficult to distinguish the nodes that form the inner wall of the element from those for the outer wall or the center fluid region. To make selection easier, Abaqus/Standard defines multiple surfaces on a thermal pipe element, as shown in *Fluid Pipe Element Library*. These surface definitions allow you to easily select the different nodes on the thermal fluid pipe elements to be used to specify a coupling to the formation or to another pipe or connector element.

Assigning a Material Definition to a Set of Fluid Pipe Elements

You must associate a material definition with each pipe element section property.

The material that is defined for the thermal fluid pipe section refers to the fluid that is flowing through the pipe. You must define the following properties for the fluid to describe its flow behavior: fluid density and viscosity. For the viscosity definition, thermal fluid pipe elements support both Newtonian and non-Newtonian fluids. The following non-Newtonian fluid models are supported: power law, Bingham Plastic, and Herschel-Bulkley models (see *Viscosity*).

For the thermal fluid pipe elements, you must also specify the following thermal properties:

- Conductivity and specific heat for the fluid
- Density, conductivity, and specific heat for the pipe wall

Input File Usage: Use the following options to specify the flow and thermal behaviors of the fluid:

**FLUID PIPE SECTION*, MATERIAL=*material name*
**MATERIAL*, NAME=*material name*
**VISCOSITY*, DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC
**DENSITY*, PORE FLUID
**CONDUCTIVITY*, PORE FLUID
**SPECIFIC HEAT*, PORE FLUID

In addition, use the following options to specify the thermal properties for the pipe wall:

**DENSITY*
**CONDUCTIVITY*
**SPECIFIC HEAT*

Multiple Fluid Flow

Abaqus allows the use of multiple fluids in an analysis, up to a maximum of four fluid types. This capability is useful in situations where the spatial distribution of the different fluid types is known (precomputed) at all times during the analysis. In other words, the type of fluid at each node in the domain is known at all times during the analysis.

The definition of multiple fluids in an analysis involves assigning a unique integer value that acts as an identifier for each fluid type, and associating each fluid identifier with a specific predefined field variable used to specify the spatial distribution of the fluid as function of time. You can utilize the fluid identifier to specify the viscosity of each fluid. Only the fluid viscosity can be different for different fluids. All other properties (for example, density, thermal) are assumed to be same for all the fluids. The field variable associated with a fluid type allows you to predefine the fluid type at a node as a function of time using amplitude definitions or as functions of both time and space utilizing user subroutine *UFIELD* or *USDFLD*.

You specify a value for each field variable, associated with a fluid identifier, at each node. It is recommended that you specify a value for the field variable (FV) such that, $0.0 \leq FV \leq 1.0$. Abaqus assigns the fluid type at an integration point to be the one for which the field variable has the maximum value at that point. If all field variables have the same numerical value at an integration point, the first fluid (fluid identifier 1) is assumed to be active.

Input File Usage: **VISCOSITY*, FLUID ID=*Fluid Id number*,DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC
**FIELD*, VARIABLE=*Field variable number*
**FLUID ID FIELD*
Fluid Id number,*Field variable number*

Using Thermal Fluid Pipe Elements in Symmetric Models

You can use thermal fluid pipe elements in models that use symmetry to reduce the model size. The thermal fluid pipe elements model circular geometry. You specify a symmetry value that is greater than 0 and less than or equal to 1. This value is interpreted as scaling of the circular pipe geometry in radians. A value of 1 corresponds to 2π radians, and a value of 0.5 corresponds to π radians. When you specify symmetry, you must scale the full model flow magnitude by the symmetry value. You do not need to apply any symmetric constraints or boundary conditions.

Input File Usage: **FLUID PIPE SECTION*, SYMMETRY=*value*

Thermal Fluid Pipe Equations

The governing equations for the fluid flow and pressure drop computation in pipes are described in *Fluid Pipe Equations*. The different choices of specifying the frictional loss characteristics for the flowing fluid in the thermal pipe are described in *Specifying the Friction Loss Behavior*. You can model additional fluid pressure losses in thermal fluid pipe elements as described in *Additional Loss Terms in Fluid Pipe Elements*.

In the thermal fluid pipe elements, the fluid temperature evolution is given by the 1D transient advection-diffusion equation:

$$\frac{d\theta}{dt} - D\frac{d^2\theta}{dx^2} + V\frac{d\theta}{dx} + \frac{h}{\rho_f C_p^f}(\theta - \theta_{wall}) = 0,$$

where:

- θ is the temperature.
- D is the fluid thermal diffusivity computed as the ratio of thermal conductivity to thermal capacity, $D = \frac{K_f}{\rho_f C_p^f}$.
  Here, ρ_f is the fluid density, K_f is the fluid thermal conductivity, and C_p^f is the specific heat of the fluid.
- V is the velocity of fluid flow.
- h is the heat convection coefficient.

Specifying the Wall Diameters for Thermal Fluid Pipe Elements

You must define the geometry of the pipe wall as a circular cross-section by specifying the inner and outer wall radii.

Input File Usage: **FLUID PIPE WALL*

Specifying the Heat Convection between the Fluid and the Pipe Wall

The heat convection coefficient is defined as

$$h = \mathbf{Nu}\frac{K_f}{D_h},$$

where

- $\mathbf{Nu}$ is the Nussell number,
- D_h
  is the hydraulic diameter defined as $D_h = 4$ (Flow Area/Flow perimeter), and
- K_f
  is the thermal conductivity of the fluid.

You can define the type of convection to determine the temperature at the interface of the pipe wall and flowing fluid (nodes 3 and 4 for the regular pipe and nodes 3, 4, 7, and 8 for the annular pipe). Newton's law of cooling is used to determine this interface temperature. The convection type determines whether a constant temperature or a constant flux method is used in this computation.

Input File Usage: Use the following option to specify a constant temperature method for convective heat transfer ($\mathbf{Nu} = \mathbf{7.54}$):

**FLUID PIPE THERMAL*, TYPE=TEMPERATURE

Use the following option to specify a constant flux method for convective heat transfer ($\mathbf{Nu = 8.235}$):

**FLUID PIPE THERMAL*, TYPE=FLUX

Specifying Initial and Prescribed Conditions

You can define an initial temperature or field distribution over the nodes of the thermal fluid pipe elements.

Input File Usage: Use one or both of the following options:

**INITIAL CONDITIONS*, TYPE=TEMPERATURE
**INITIAL CONDITIONS*, TYPE=FIELD

Specifying Loads and Boundary Conditions

Thermal fluid pipe elements allow for the specification of pressure boundary conditions and volumetric flow rates at the nodes. At a particular node, you can specify either a pressure or flow rate but not both. You can also specify a gravity load on the thermal fluid pipe element to determine the hydrostatic head at the nodes. You can specify the fluid temperature at the inlet or outlet and at the pipe wall.

Input File Usage: Use the following option to specify the pressure at the inlet or outlet:

**BOUNDARY*
node or node set, *8*, *8*, *magnitude*

Use the following option to specify the flow rate at the inlet or outlet:

**CFLOW*
node or node set, , *magnitude*

Use the following option to specify the temperature at the inlet, outlet, or pipe wall:

**BOUNDARY*
node or node set, *11*, *11*, *magnitude*

Use the following option to specify a gravity load on the fluid pipe element:

**DLOAD*
element or element set, , *GRAV*, *gravity constant*, *comp1*, *comp2*, *comp3*

Slurry Fluid Pipe Elements

Products: Abaqus/Standard

References:

- *Fluid Pipe Element Library*
- **FLUID PIPE SECTION*
- **FLUID PIPE FLOW LOSS*
- **FLUID PIPE THERMAL*
- **FLUID PIPE WALL*

Overview

Slurry fluid pipe and thermal-slurry fluid pipe elements in Abaqus/Standard allow you to simulate the pressure loss due to viscous and gravity effects in a fluid pipe network carrying a slurry, including the temperature field in the fluid flowing in the pipe, heat exchange between the fluid and the pipe, and heat transfer through the walls of the pipe.

A slurry is a mixture of a carrier fluid and solid proppant particles. Slurry fluid pipe elements model the flow field using a pure pressure formulation, based on Bernoulli's equation for steady-state flow of a single-phase, incompressible fluid through a fully filled pipe with a constant cross-sectional area. The fractional increase in pressure loss due to the presence of proppants in the carrier fluid is modeled using the Turian approach. The thermal model includes convection and diffusion of heat in the fluid, as well as conduction of heat along the length and radial (thickness) directions in the pipe.

Typical Applications

Slurry fluid pipe elements are used to simulate:

- the flow of a slurry through a pipe or network of pipes to determine pressure drops and flow rates, and
- the temperature field in the flowing fluid and the pipe walls.

These elements can be used in a geostatic or coupled pore fluid diffusion/stress analysis (see *Geostatic Stress State* and *Coupled Pore Fluid Diffusion and Stress Analysis*). They can also be used to model one-dimensional wellbores in certain geotechnical applications.

Choosing an Appropriate Element

Several fluid pipe elements with slurry modeling capabilities are available. They can be broadly classified as follows:

- Regular slurry fluid pipe elements (FP2D2C and FP3D2C)
- Regular thermal slurry fluid pipe elements (FP2D2Q and FP3D2Q)
- Annular thermal slurry fluid pipe elements (FP2D2AQ and FP3D2AQ)

For two-dimensional analyses, use element type FP2D2C, FP2D2Q, or FP2D2AQ. For three-dimensional analyses, use element type FP3D2C, FP3D2Q, or FP3D2AQ. For a detailed discussion of the differences between the two classes (*regular* versus *annular*) of fluid pipe elements, see *Choosing an Appropriate Element* in *Thermal Fluid Pipe Elements*. For defining surfaces on regular and annular fluid pipe elements refer to *Fluid Pipe Element Library*.

Using Slurry Fluid Pipe Elements in Symmetric Models

You can use slurry fluid pipe elements and thermal-slurry fluid pipe elements in models that use symmetry to reduce the model size. These elements model circular geometry. You can specify a symmetry value that is greater than 0 and less than or equal to 1. This value is interpreted as scaling of the circular pipe geometry in radians.

A value of 1 corresponds to 2π radians, and a value of 0.5 corresponds to π radians. When you specify symmetry, you must scale the full model flow magnitude by the symmetry value. You do not need to apply any symmetric constraints or boundary conditions.

Input File Usage: **FLUID PIPE SECTION*, SYMMETRY=*value*

Slurry Fluid Pipe Equations

Fluid Pipe Equations describes the governing equations for the fluid flow and pressure drop computation for fluids in pipes. *Specifying the Friction Loss Behavior* describes the different choices of specifying the frictional loss characteristics for the flowing fluid in the thermal pipe. You can model additional fluid pressure losses in thermal fluid pipe elements (see *Additional Loss Terms in Fluid Pipe Elements*). The governing equations to model temperature evolutions are described in *Thermal Fluid Pipe Equations*. When using the thermal slurry fluid pipe elements, you can specify the pipe wall section and convection behavior as described in *Specifying the Wall Diameters for Thermal Fluid Pipe Elements* and *Specifying the Heat Convection between the Fluid and the Pipe Wall*.

Abaqus/Standard models the slurry as a mixture of proppant particles and a carrying fluid. The proppant concentration, C_v, is defined as the volume ratio of the solid phase to that of the slurry phase. The fluid pipe elements assume that the proppant particles flow with the mean flow velocity of the slurry. Abaqus/Standard models gravity settling indirectly by including the effects of settling on the friction loss in the pipe (as described later in this section). The governing equation for proppant concentration is the advection equation for a scalar field, given by

$$\frac{\partial C_v}{\partial t} + \frac{\partial (V C_v)}{\partial x} = 0,$$

where V is the mean flow velocity of the slurry in the pipe. Abaqus/Standard uses a Petrov-Galerkin scheme to discretize the governing equation for the slurry concentration.

When slurry flows through the tubing, pipe, or annulus, the flow conditions might change and lead to heterogeneous slurry flow and settling of solids. This can be observed in slurry transport through long horizontal sections at flow velocities too low to maintain the solids in suspension. Under horizontal flow conditions and the influence of gravity, significant gradation in the concentration of solids can develop in the form of various slurry flow regimes (for example, heterogeneous, saltation, or sliding bed). The presence of solid particles in the slurry generates additional momentum transfer between the particles and pipe wall, in addition to the transfer between the carrier fluid and pipe wall. The net effect of the additional transfer mechanism is an increase in pressure drop. Based on the concept of additive pressure drop, the fractional increase in pressure gradient due to the presence of solid particles can be added to that produced by interaction of the carrier fluid with the pipe wall:

$$\Delta P_{f,m} = \Delta P_{f,w} + \Delta P_{f,s},$$

where:

- $\Delta P_{f,m}$ is the pressure drop of the mixture,
- $\Delta P_{f,w}$ is the pressure drop due to the flow of the carrier fluid alone, and
- $\Delta P_{f,s}$ is the pressure drop due to the flow of solid particles.

Based on many experimental observations, an extended pressure loss definition was proposed by Turian and Yuan (1977), to account for various flow regimes often prevailing in the transport of slurry depending upon flow conditions. The pressure gradient in each flow regime can be correlated using an equation of the form:

$$f_m - f_w = A_1 {C_v}^{A_2} {f_w}^{A_3} {C_D^*}^{A_4} [Fr]^{A_5},$$

where:

- C_v is the concentration of proppant particles in the slurry, defined as the volume ratio of the solid phase to that of the slurry phase (subsequently simply referred to as the slurry concentration);
- f_w is the friction factor due to the carrier fluid flow;
- C_D^* is the drag coefficient of a single sphere settling at terminal velocity in an unbounded quiescent fluid; and
- Fr is the pipe Froude number modified to account for particle density ρ_s and is defined as $\left[\frac{V^2}{gD_P\left(\frac{\rho_s}{\rho_w}-1\right)}\right]$, where V is the flow velocity, g is gravity, and D_P is the particle diameter.

Based on the work of Turian and Yuan (1977), the values of these coefficients were defined for each of the slurry flow regimes, namely sliding bed (Regime 0), saltation (Regime 1), heterogeneous suspension (Regime 2) and homogeneous suspension (Regime 3):

Sliding Bed (Regime 0 - R_0):

$$f_m - f_w = 12.13 {C_v}^{0.7389} {f_w}^{0.7717} {C_D^*}^{-0.4054} [Fr]^{-1.096}$$

Saltation (Regime 1 - R_1):

$$f_m - f_w = 107.1 {C_v}^{1.018} {f_w}^{1.046} {C_D^*}^{-0.4213} [Fr]^{-1.354}$$

Heterogeneous suspension (Regime 2 - R_2):

$$f_m - f_w = 30.11 {C_v}^{0.868} {f_w}^{1.200} {C_D^*}^{-0.1677} [Fr]^{-0.6938}$$

Homogeneous suspension (Regime 3 - R_3):

$$f_m - f_w = 8.538 {C_v}^{0.5024} {f_w}^{1.428} {C_D^*}^{0.1516} [Fr]^{-0.3531}$$

Demarcation of the slurry flow regimes is accomplished by ensuring that any two regimes are contiguous at their common boundary. This implies that any two regime correlation equations must be satisfied simultaneously. For example, the boundary demarcating the sliding bed and saltation regimes must lie along the locus of the equation for regimes $R_0 = R_1$. The transition number demarcating the boundary between the sliding bed (R_0) and saltation (R_1) regimes can be written as

$$R_{01} = \frac{Fr}{4679 {C_v}^{1.083} {f_w}^{1.064} {C_D^*}^{-0.0616}}.$$

Similarly, the equations for other regime transitions can be written as

$$R_{02} = \frac{Fr}{0.1044 {C_v}^{-0.3225} {f_w}^{-1.065} {C_D^*}^{-0.5906}}$$

$$R_{03} = \frac{Fr}{1.6038 {C_v}^{0.3183} {f_w}^{-0.8837} {C_D^*}^{-0.7496}}$$

$$R_{12} = \frac{Fr}{6.8359 {C_v}^{0.2263} {f_w}^{-0.2334} {C_D^*}^{-0.3840}}$$

$$R_{13} = \frac{Fr}{12.522 {C_v}^{0.5153} {f_w}^{-0.3820} {C_D^*}^{-0.5724}}$$

$$R_{23} = \frac{Fr}{40.38 {C_v}^{1.075} {f_w}^{-0.6700} {C_D^*}^{-0.9375}}$$

An important aspect of implementing the approach developed by Turian et al., is a process by which the flow regime can be readily established. It is possible to identify the applicable regime for a given set of flowing conditions just from knowledge of the transition numbers R_{ij}. For the transition numbers R_{ij}, where $i < j$, the transition number R_{ij} increases monotonically as the flow velocity increases. At low flow velocities, $R_{ij} < 1$ and eventually passes through the value 1.0 with increasing flow velocity. This indicates a transition out of Regime *i*. For any combination of average flow velocity, V, and particle diameter D_p, the flow regime is fixed according to the constraints: $R_{ij} < 1$, the regime is not *j*; and $R_{ij} > 1$, the regime is not *i*. For further details on the algorithm, refer to Turian and Yuan (1977).

After the drag coefficient, friction factor, and pipe Froude number are calculated, the transition numbers are calculated and tested to identify the correct slurry flow regime. Once the slurry flow regime is established, the slurry friction factor and fractional loss due to solids are determined.

Assigning a Material Definition to a Set of Fluid Pipe Elements

You must associate a material definition with each pipe element section property.

The material that is defined for the thermal-slurry fluid pipe section refers to the slurry that is flowing through the pipe and the thermal properties for the pipe wall. You must define the fluid density and the viscosity of the slurry to describe its flow behavior. The density of the slurry can be specified to be a function of the slurry concentration. In this case, the density is computed using the density of the fluid and the solid proppant particles, and the slurry concentration, using a rule of mixtures. The viscosity of the slurry can be specified as a function of the slurry concentration directly in tabular form. For the viscosity definition, slurry fluid pipe elements support both Newtonian and non-Newtonian fluids. The following non-Newtonian fluids are supported: power law, Bingham Plastic, and Herschel-Bulkley models (see *Viscosity*).

In addition, for the thermal-slurry fluid pipe elements, you must specify the following thermal properties:

- Conductivity and specific heat of the slurry. The conductivity can optionally be specified as a function of the slurry concentration (as well as temperature and field variables, if necessary) directly in tabular form. The specific heat can also be specified to be dependent on the slurry concentration. In this case the specific heat of the slurry is computed using a rule of mixtures based on the specific heat of the carrying fluid, the specific heat of the solid proppant particles, and the slurry concentration.

Input File Usage: Use the following options to specify the modeling of slurry flow and thermal behaviors of the fluid:

**FLUID PIPE SECTION*, MATERIAL=*material name*
**MATERIAL*, NAME=*material name*
**VISCOSITY*, DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC, SLURRY
**DENSITY*, PORE FLUID, SLURRY
**CONDUCTIVITY*, PORE FLUID, SLURRY
**SPECIFIC HEAT*, PORE FLUID, SLURRY

In addition, use the following options to specify the thermal properties for the pipe wall:

**DENSITY*
**CONDUCTIVITY*
**SPECIFIC HEAT*

Multiple Fluid Flow

Abaqus allows the use of multiple fluids in an analysis, up to a maximum of four fluid types. This capability is useful in situations where the spatial distribution of the different fluid types is known (precomputed) at all times during the analysis. In other words, the type of fluid at each node in the domain is known at all times during the analysis.

The definition of multiple fluids in an analysis involves assigning a unique integer value that acts as an identifier of each fluid type, and associating each fluid identifier with a specific predefined field variable used to specify the spatial distribution of the fluid as function of time. You can utilize the fluid identifier to specify the viscosity of each fluid. Only the fluid viscosity can be different for different fluids. All other properties (that is, density and thermal) are assumed to be same for all the fluids. The field variable associated with a fluid type allows you to predefine the fluid type at a node as a function of time using amplitude definitions or as functions of both time and space utilizing user subroutine *UFIELD* or *USDFLD*.

You specify a value for each field variable, associated with a fluid identifier, at each node. It is recommended that you specify a value for the field variable (FV) such that $0.0 \leq FV \leq 1.0$. Abaqus assigns the fluid type at an integration point to be the one for which the field variable has the maximum value at that point. If all field variables have the same numerical value at an integration point, the first fluid (fluid identifier equal to 1) is assumed to be active.

Input File Usage:
**VISCOSITY*, FLUID ID=*Fluid Id number*,DEFINITION=NEWTONIAN or POWER LAW or HERSCHEL-BULKLEY or BINGHAM PLASTIC
**FIELD*, VARIABLE=*Field variable number*
**FLUID ID FIELD*
Fluid Id number,*Field variable number*

Specifying Initial and Prescribed Conditions

You can define an initial slurry concentration, an initial temperature, and a field distribution over the nodes of the slurry fluid pipe element and thermal-slurry fluid pipe element.

Input File Usage:
Use one or both of the following options:

**INITIAL CONDITIONS*, TYPE=TEMPERATURE
**INITIAL CONDITIONS*, TYPE=FIELD
**INITIAL CONDITIONS*, TYPE=SLURRYVF

Specifying Loads and Boundary Conditions

Slurry fluid pipe elements allow for the specification of pressure boundary conditions and volumetric flow rates at the nodes. At a particular node, you can specify either a pressure or flow rate but not both. You can also specify a gravity load on the slurry fluid pipe element and thermal-slurry fluid pipe element to determine the hydrostatic head at the nodes. For the thermal-slurry fluid pipe elements, you can specify the fluid temperature at the inlet or outlet and at the pipe wall. You can specify the slurry concentration at the inlet or outlet of the node that captures the fluid pressure loss.

Input File Usage:
Use the following option to specify the pressure at the inlet or outlet:

**BOUNDARY*
node or node set, *8*, *8*, *magnitude*

Use the following option to specify the flow rate at the inlet or outlet:

**CFLOW*
node or node set, , *magnitude*

Use the following option to specify the temperature at the inlet, outlet, or pipe wall:

**BOUNDARY*
node or node set, *11*, *11*, *magnitude*

Use the following option to specify the slurry volume fraction at the inlet, outlet, or pipe wall:

**BOUNDARY*
node or node set, *31*, *31*, *magnitude*

Use the following option to specify a gravity load on the fluid pipe element:

**DLOAD*
element or element set, , *GRAV*, *gravity constant*, *comp1*, *comp2*, *comp3*

References

- Turian, R. M., , and T-F. Yuan, "Flow of Slurries in Pipelines," *American Institute of Chemical Engineers Journal*, vol. 23, no. 3, 232-243, 1977.

Fluid Pipe Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the fluid pipe elements available in Abaqus/Standard.

References:

- *Fluid Pipe Elements*
- *Thermal Fluid Pipe Elements*
- **FLUID PIPE SECTION*
- **FLUID PIPE FLOW LOSS*
- **FLUID PIPE THERMAL*
- **FLUID PIPE WALL*

Element Types

2D Pore Pressure Element

FP2D2	2-node linear

Active Degrees of Freedom

8

Additional Solution Variables

None.

2D Coupled Temperature—Pore Pressure Elements

FP2D2T	6-node linear
FP2D2AT	8-node linear, annular

Active Degrees of Freedom

8 at the first two nodes of element connectivity

11 at all nodes

Additional Solution Variables

None.

2D Coupled Slurry—Pore Pressure Element

FP2D2C	2-node linear

Active Degrees of Freedom

8

31

Additional Solution Variables

None.

2D Coupled Temperature—Slurry—Pore Pressure Elements

FP2D2Q	6-node linear
FP2D2AQ	8-node linear, annular

Active Degrees of Freedom

8 at the first two nodes of element connectivity

11 at all nodes

31 at the first two nodes of element connectivity

Additional Solution Variables

None.

3D Pore Pressure Element

FP3D2	2-node linear

Active Degrees of Freedom

8

Additional Solution Variables

None.

3D Coupled Temperature—Pore Pressure Elements

FP3D2T	6-node linear
FP3D2AT	8-node linear, annular

Active Degrees of Freedom

8 at the first two nodes of element connectivity

11 at all nodes

Additional Solution Variables

None.

3D Coupled Slurry—Pore Pressure Element

FP3D2C	2-node linear

Active Degrees of Freedom

8

31

Additional Solution Variables

None.

3D Coupled Temperature—Slurry—Pore Pressure Elements

FP3D2Q	6-node linear
FP3D2AQ	8-node linear, annular

Active Degrees of Freedom

8 at the first two nodes of element connectivity

11 at all nodes

31 at the first two nodes of element connectivity

Additional Solution Variables

None.

Nodal Coordinates Required

2D: *X*, *Y*

3D: *X*, *Y*, *Z*

Element Property Definition

The fluid pipe elements FP2D2, FP2D2C, FP3D2, and FP3D2C are defined by two nodes, which cannot be coincident (a zero-length pipe is not allowed).

The thermal fluid pipe elements FP2D2T and FP3D2T have six nodes. Two nodes define the fluid pressure and temperature, and four nodes define the wall temperatures. The nodes that define the pipe wall are coincident with the nodes that define the pore pressure degrees of freedom.

The thermal-slurry fluid pipe elements FP2D2Q and FP3D2Q have six nodes. Two nodes define the fluid pressure, slurry, and temperature; and four nodes define the wall temperatures.The nodes that define the pipe wall are coincident with the nodes that define the pore pressure degrees of freedom.

The annular thermal fluid pipe elements FP2D2AT and FP3D2AT have eight nodes. Two nodes define the fluid pressure and temperature, and the remaining six nodes define the wall temperatures.

The annular thermal-slurry fluid pipe elements FP2D2AQ and FP3D2AQ have eight nodes. Two nodes define the fluid pressure, slurry, and temperature; and the remaining six nodes define the wall temperatures.

You must associate a fluid pipe section with a set of fluid pipe elements.

Input File Usage: **FLUID PIPE SECTION*, ELSET=*name*

Element-Based Loading

Distributed Loads

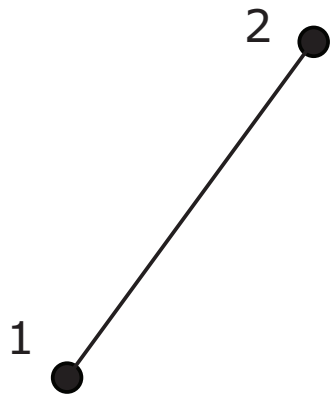
Distributed loads are specified as described in *Distributed Loads*.

Load ID (*DLOAD): GRAV
Units: LT^{-2}
Description: Gravity loading in specified direction (magnitude is input as acceleration).

Element Output

FPDPRESS	Pressure drop across the element.
FPMFL	Mass flow rate through the element.
FPFLVEL	Velocity of the fluid flowing through the element.

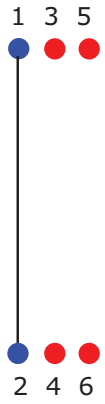
Node Ordering on Elements



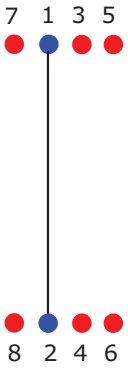
FP2D2 FP3D2
FP2D2C FP3D2C



FP2D2T FP3D2T



FP2D2Q FP3D2Q



FP2D2AT FP3D2AT



FP2D2AQ FP3D2AQ

The following table displays the surface definition for coupled thermal-pore pressure fluid pipe elements (FP2D2T, FP3D2T) and coupled thermal-pore pressure slurry fluid pipe elements (FP2D2Q, FP3D2Q).

Face ID	Nodes
Face 1	5 – 6
Face 2	3 – 4

Face ID	Nodes
Face 3	1 – 2
Face 4	5
Face 5	3
Face 6	1
Face 7	6
Face 8	4
Face 9	2

The following table displays the surface definition for coupled thermal-pore pressure annulus fluid pipe elements (FP2D2AT, FP3D2AT) and coupled thermal-pore pressure slurry fluid pipe annulus elements (FP2D2AQ, FP3D2AQ).

Face ID	Nodes
Face 1	5 – 6
Face 2	3 – 4
Face 3	1 – 2
Face 4	5
Face 5	3
Face 6	1
Face 7	6
Face 8	4
Face 9	2
Face 10	7 - 8
Face 11	7
Face 12	8

Gasket Elements

In this section:

- *About Gasket Elements*
- *Choosing a Gasket Element*
- *Including Gasket Elements in a Model*
- *Defining the Gasket Element's Initial Geometry*
- *Defining the Gasket Behavior Using a Material Model*
- *Defining the Gasket Behavior Directly Using a Gasket Behavior Model*
- *Defining a Thermal Interaction for a Gasket Element*
- *Two-Dimensional Gasket Element Library*
- *Three-Dimensional Gasket Element Library*
- *Axisymmetric Gasket Element Library*

About Gasket Elements

Overview

Gasket modeling consists of:

- choosing the appropriate gasket element type (*Choosing a Gasket Element*);
- including the gasket elements in a finite element model (*Including Gasket Elements in a Model*);
- defining the initial geometry of the gasket (*Defining the Gasket Element's Initial Geometry*);
- defining the gasket behavior with either a material model (*Defining the Gasket Behavior Using a Material Model*) or a gasket behavior model (*Defining the Gasket Behavior Directly Using a Gasket Behavior Model*); and
- defining the thermal interaction between the top and bottom surfaces of a coupled temperature-displacement gasket element(*Defining a Thermal Interaction for a Gasket Element*).

Abaqus/Standard offers a library of gasket elements to model the behavior of gaskets.

References:

Motivation for Gasket Elements

Gaskets are constructed in many ways and from many materials. Some types of gaskets consist of several layers of preformed metal, possibly with thin elastomeric coatings or elastomeric inserts (see *Figure 1*). Others use plastics together with elastomeric inserts.

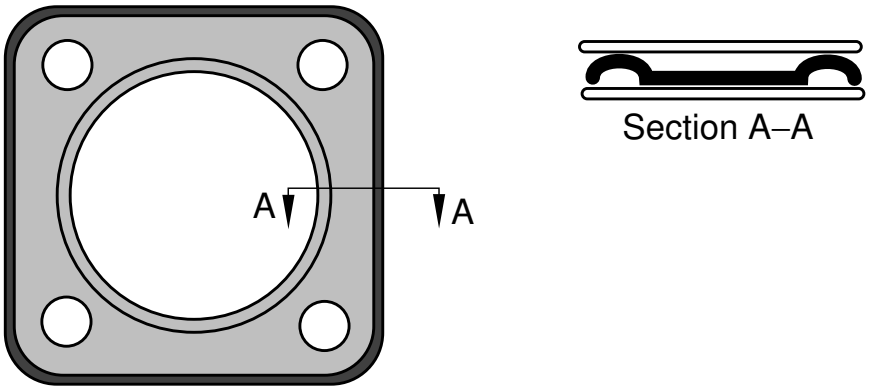


Figure 1: Typical gasket consisting of several layers of preformed metal.

Gaskets are usually very thin and act as sealing components between structural components. They are carefully designed to provide appropriate pressure-closure behaviors through their thickness (the thin direction of the gaskets) so that they maintain their sealing action as the components undergo deformations due to thermal and mechanical loads. It is difficult to use solid continuum elements to model the through-thickness behavior of gaskets with the material library available. Therefore, Abaqus/Standard offers a variety of gasket elements that have through-thickness behaviors specifically designed for the study of gaskets.

The gasket behavior models are separate from the models in the material library and assume that the thickness-direction, transverse shear, and membrane behaviors are uncoupled (see *Defining the Gasket Behavior Directly Using a Gasket Behavior Model* for details). For a gasket behavior that is not readily addressed by these

special behavior models, such as occurs when coupled behaviors or through-thickness tensile behavior must be considered, Abaqus/Standard provides a versatile alternative by allowing a gasket element to use either a built-in or user-defined material model (see *Defining the Gasket Behavior Using a Material Model* for details).

Spatial Representation of a Gasket Element

Figure 2 demonstrates the key geometrical features that are used to define gasket elements. Gasket elements are composed of two surfaces separated by a thickness. The relative motion of the bottom and top surfaces measured along the thickness direction to the gasket quantifies the thickness-direction (local 1-direction) behavior of the gasket element. The relative change in position of the bottom and top surfaces measured in the plane orthogonal to the thickness direction quantifies the transverse shear behavior of the gasket element. The stretching and shearing of the midsurface of the element (the surface halfway between the bottom and top surfaces) quantifies the membrane behavior of the gasket element.

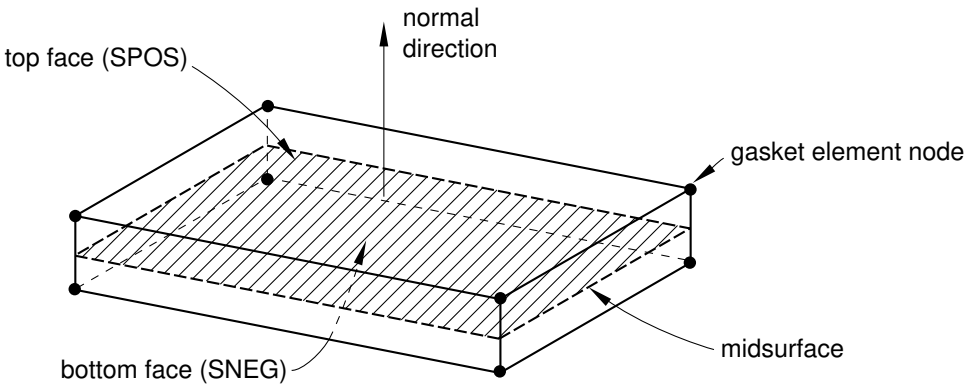


Figure 2: Spatial representation of a gasket element.

Local Behavior Directions Defined at the Integration Points

The thickness direction defined at the integration points of gasket elements constitutes the local 1-direction. The transverse shear behavior is defined in the local 1–2 and 1–3 planes. The membrane behavior is defined in the 2–3 plane. The local 2- and 3-directions are not defined for elements that have nodes with only one degree of freedom because these elements consider only the thickness-direction behavior of a gasket. The local directions are used to specify the gasket behavior and for output of all quantities that describe the current deformation state of a gasket. Abaqus/Standard computes the local directions by default. You can also define them for some element types.

Default Local Directions

Abaqus/Standard computes the local 1-direction as explained in *Defining the Gasket Element's Initial Geometry*.

For two-dimensional and axisymmetric gasket elements, the local 2-direction is defined so that the cross product between the local 1- and 2-directions gives the out-of-plane direction (see *Figure 3*).

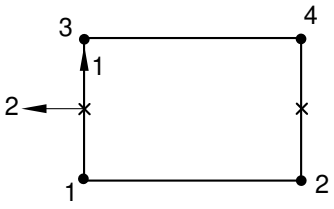


Figure 3: Local directions for two-dimensional and axisymmetric gasket elements.

For three-dimensional area and three-dimensional link elements, the local 2- and 3-directions are normal to the local 1-direction (see *Figure 4*) and are defined by the standard Abaqus convention for local directions on surfaces in space (see *Conventions*).

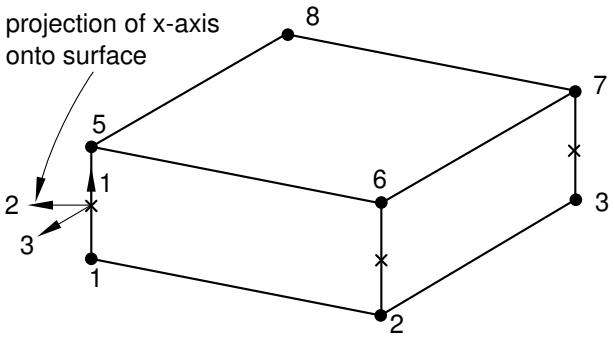


Figure 4: Local directions for three-dimensional area and three-dimensional link gasket elements.

For three-dimensional line elements, the local 2-direction is obtained by the projection of the tangent to the midsurface of the element onto the plane orthogonal to the local 1-direction (see *Figure 5*). The local 3-direction is then obtained by the cross product of the local 1- and 2-directions.

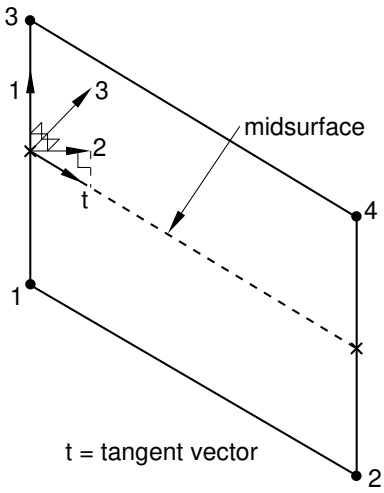


Figure 5: Local directions for three-dimensional line gasket elements.

Specifying the Local Directions

You can define the local 1-direction as explained in *Defining the Gasket Element's Initial Geometry*. The local 2- and 3-directions can be defined using local orientations (*Orientations*) for three-dimensional area and three-dimensional link elements that consider transverse shear and membrane deformations.

Input File Usage: Use the following option to associate a local orientation with a particular gasket element set:

**GASKET SECTION*, ELSET=*name*, ORIENTATION=*name*

Abaqus/CAE Usage: Property module: **Assign->Material Orientation**

Procedures with Which Gasket Elements Are Allowed

Gasket elements can be used in static, static perturbation, quasi-static, dynamic, and frequency analyses. Coupled temperature-displacement gasket elements can be also used in coupled thermal-stress analysis. However, gasket elements are assumed to have no mass; therefore, the density cannot be defined for gasket elements.

Choosing a Gasket Element

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Gasket Elements*
- *Two-Dimensional Gasket Element Library*
- *Three-Dimensional Gasket Element Library*
- *Axisymmetric Gasket Element Library*
- *Gaskets*

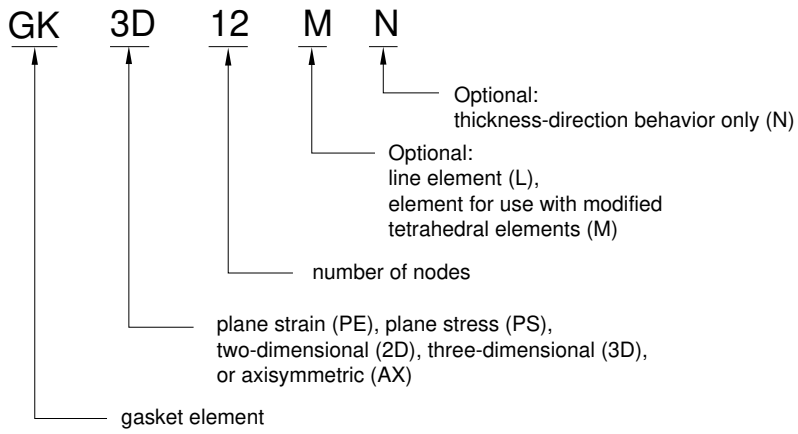
Overview

The Abaqus/Standard gasket element library includes:

- elements for two-dimensional analyses;
- elements for three-dimensional analyses;
- elements for axisymmetric analyses;
- elements that account for the thickness-direction behavior of gaskets only; and
- elements that account for the thickness-direction, membrane, and transverse shear behaviors of gaskets.

Naming Convention

The gasket elements used in Abaqus/Standard are named as follows:



For example, GKPE4 is a 4-node, plane strain gasket element that accounts for thickness-direction, membrane, and transverse shear behaviors.

Elements for General Use Versus Elements with Thickness-Direction Behavior Only

Abaqus/Standard offers two classes of gasket elements. In both classes material properties can be specified by either special gasket behavior models or built-in material models, including user-defined materials (see *Defining the Gasket Behavior Directly Using a Gasket Behavior Model* and *Defining the Gasket Behavior Using a Material Model*). The first class is a collection of gasket elements that have all displacement degrees of freedom active

at their nodes. These elements are necessary when the membrane and/or transverse shear behavior of the gasket is of importance (see *Figure 1*). The thickness-direction, transverse shear, and membrane behaviors can be defined as uncoupled behaviors only, when the elements are used in conjunction with special gasket behavior models. In some cases the membrane effects are only secondary; in such cases it is possible to model only the thickness-direction and transverse shear behaviors. These elements are suited for analyses where both thickness-direction behavior and frictional effects are important.

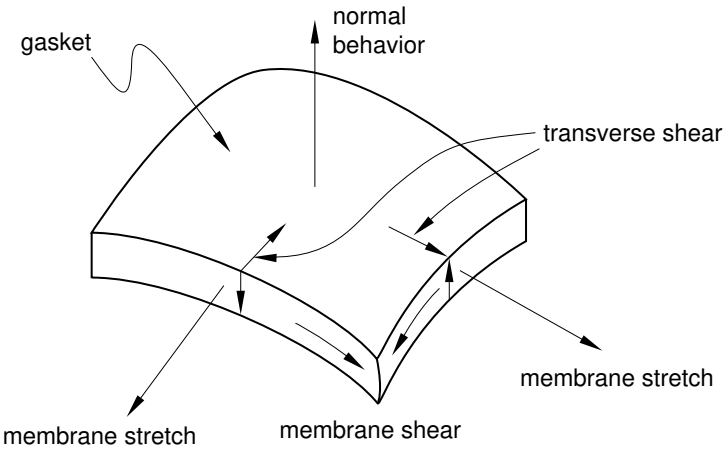


Figure 1: Different deformation modes of gaskets.

In the second class of gasket elements deformation is measured only in the thickness direction. The response of the gasket to any other deformation mode is ignored. The nodes of these elements have only one displacement degree of freedom, which lies in the thickness direction of the gasket. This class of elements is intended as a means to reduce the computational cost of an analysis when the thickness-direction behavior of the gasket is the only behavior of importance. This behavior can be specified easily in terms of pressure in the gasket versus gasket closure. Frictional forces cannot be transmitted by such elements, and any thermal expansion or stretching of the gasket in its plane is not accounted for.

Elements for Two-Dimensional, Three-Dimensional, and Axisymmetric Analyses

For both classes of gasket elements Abaqus/Standard offers a choice of two-dimensional, three-dimensional, and axisymmetric elements. Plane stress and plane strain elements are provided for two-dimensional analyses to represent thin gaskets or thick gaskets in the out-of-plane direction, respectively. Axisymmetric gasket elements are provided for cases where the geometry and loading of the structure are axisymmetric.

Abaqus/Standard offers 2-node or link elements for two-dimensional, three-dimensional, and axisymmetric analyses; three-dimensional line elements; and a three-dimensional 12-node element for use with modified tetrahedral elements. These elements have specific characteristics that are useful when modeling gaskets.

Link Elements

Because link gasket elements have two nodes, their geometry defines only one dimension of the gasket—the through-thickness dimension. A link gasket element might typically be used to model a washer used under a bolt, when the bolt itself is modeled with a truss element. For two-dimensional and three-dimensional link elements the cross-section of the gasket is undetermined. For axisymmetric link elements the width of the element is undetermined. The reduction in dimensionality of these elements offers flexibility in the specification of the gasket behavior and can prove to be very efficient in some cases; see *Defining the Gasket Behavior Directly Using a Gasket Behavior Model* for further details.

Three-Dimensional Line Elements

Three-dimensional line gasket elements are typically used to model narrow, thicker features in gaskets, such as an elastomeric insert around a hole. Since they are used in three-dimensional analyses, their width is undetermined from the element's geometry. This reduction in dimensionality offers flexibility in the specification of the gasket behavior and can prove to be very efficient in some cases; see *Defining the Gasket Behavior Directly Using a Gasket Behavior Model* for further details.

12-Node Elements for Use with Modified Tetrahedral Elements

The 12-node gasket elements have the same contact properties as the modified 10-node tetrahedra; these elements have consistent nodal forces at the corner and midside nodes. They are primarily intended for use with the modified tetrahedral elements but can also be used in conjunction with other solid continuum elements by using contact pairs. In the latter case the solution may be noisy for badly mismatched meshes.

Including Gasket Elements in a Model

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Gasket Elements*
- *Choosing a Gasket Element*
- *About Contact Interactions*
- *General Multi-Point Constraints*
- *Gaskets*

Overview

Gasket elements:

- are used to model gaskets and other seals between two components, each of which may be deformable or rigid; and
- are connected to the adjacent components by sharing nodes, by using surface-based tie constraints, by using MPCs type TIE or PIN, or by using contact pairs.

This section discusses the techniques that are available to discretize gaskets and assemble them in a model representing several components, such as an internal combustion engine. The methods described all apply to gasket elements that have all displacement degrees of freedom active at their nodes. Typically they also apply to gasket elements with only thickness-direction behavior; exceptions are discussed later in this section.

Discretizing Gaskets Using Gasket Elements

Gaskets are generally manufactured as independent components. The gasket behavior is usually measured by performing a compression experiment on the gasket. In this case the gasket can be discretized as a single layer of gasket elements.

Gaskets are sometimes made of several layers of materials. If the behavior of the gasket is obtained by compression testing of the entire gasket, the gasket can again be discretized as a single layer of gasket elements. However, if the behavior of the gasket is obtained by compression testing of each layer constituting the gasket, the gasket can be discretized with a corresponding set of layers of gasket elements.

Discretizing Gaskets with Multiple Layers

If layers of gasket elements are used in the thickness direction and these layers do not have the same element layout in the plane of the gasket, use surface-based tie constraints, mesh refinement MPCs, or tied contact pairs to connect the different layers of the gasket. If tied contact pairs are used, assign a positive value to the adjustment zone depth, a, for the contact pairs (see *Contact Initialization for Contact Pairs in Abaqus/Standard*) so that all secondary nodes are properly tied at the beginning of the analysis.

Assembling Gaskets to Other Components in a Model

The easiest method to connect gasket elements that use all displacement components at their nodes to other components in a model is to define the mesh so that the gasket elements can share nodes with the elements on the surfaces of the adjacent components. More generally, when the gasket mesh is not matched to the meshing of the surfaces of the adjacent components or when the gasket elements that consider only thickness-direction behavior are used, gasket elements can be connected to other components by using contact pairs.

Connecting Gaskets to Other Components by Using Contact Pairs or Surface-Based Constraints

Gaskets are usually composed of materials that are softer than the materials that compose the neighboring components. In addition, the discretization of gaskets will usually be finer than the discretization of neighboring parts. These two facts suggest that the contacting surfaces of a gasket should be the secondary surfaces and that the contacting surfaces of neighboring parts should be the main surfaces. The second consideration also suggests that mismatched meshes will often be used in analyses involving gaskets. If mismatched meshes are used, the pressure distribution on a compressed gasket may not be predicted accurately; submodeling (*About Submodeling*) may be required to obtain accurate local results. Two techniques are available to connect gasket elements to other parts in the model when surface-based constraints are used.

Using a Regular Contact Pair and a Tied Contact Pair or a Surface-Based Constraint

This technique is required when the gasket membrane behavior is not defined. Use a tied contact pair (*Defining Tied Contact in Abaqus/Standard*) or a tie constraint (*Mesh Tie Constraints*) on one side of the gasket and a regular contact pair on the other side, as shown in *Figure 1*.

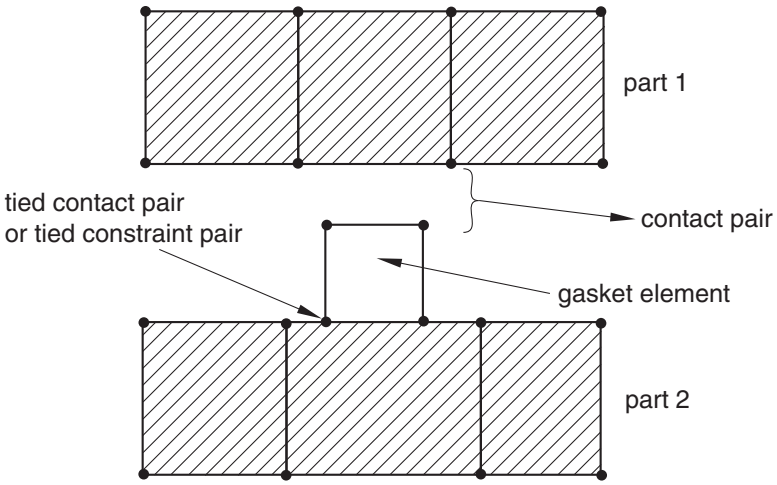


Figure 1: Connecting gaskets to other parts using contact pairs.

Because a regular contact pair is used on one side of the gasket, tensile stresses cannot develop in the gasket thickness direction should the components surrounding the gasket be pulled apart.

Assign a positive value to the adjustment zone depth, *a*, for the tied contact pair (see *Contact Initialization for Contact Pairs in Abaqus/Standard*) or, if necessary, specify a position tolerance for the tie constraint (see *Mesh Tie Constraints*) so that all secondary nodes are properly tied at the beginning of the analysis. This technique allows for frictional slip on only one side of the gasket.

Using a Regular Contact Pair and a Contact Pair That Does Not Allow Separation

This technique allows for frictional slip to be transmitted on both sides of the gasket. It is recommended when membrane behavior is defined for the gasket since it allows for the gasket membrane to stretch or contract as a result of frictional effects considered on both sides of the gasket. A contact pair or a constraint pair that does not allow for separation of the surfaces (*Contact Pressure-Overclosure Relationships*) should be used on one side of the gasket and a regular contact pair on the other, as shown in *Figure 2*.

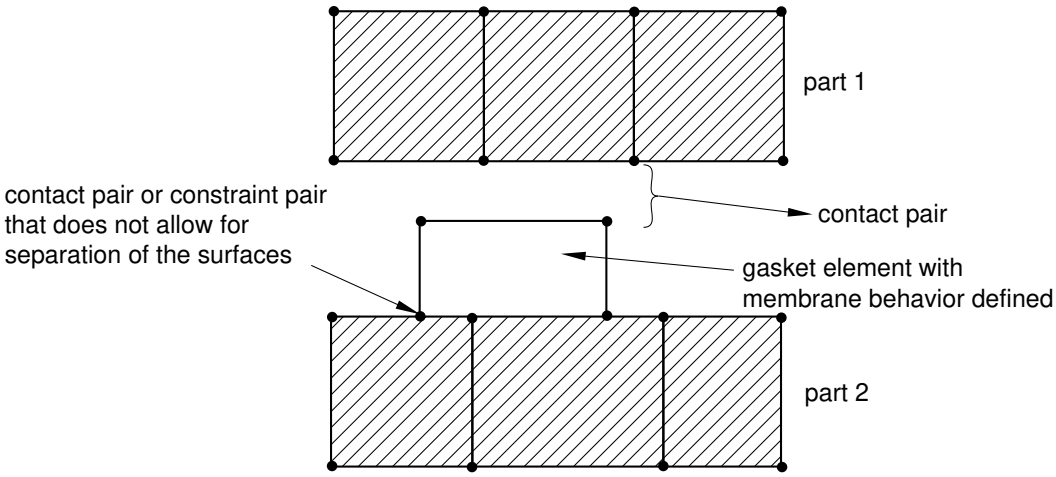


Figure 2: Connecting gaskets to other parts when the gasket membrane behavior is defined.

Assign a positive value to the adjustment zone depth, *a*, for the contact pair (see *Contact Initialization for Contact Pairs in Abaqus/Standard*) so that the surfaces are in contact at the beginning of the analysis. Use the no separation contact pressure-overclosure relationship (see *Contact Pressure-Overclosure Relationships*) so that these surfaces do not separate during the analysis. This technique will prevent rigid body modes of the gasket in its thickness direction. You may still need to prevent rigid body modes in the plane of the gasket until frictional forces develop between the gasket and the adjacent components.

Having Gasket Elements Share Nodes with Other Elements

When the gaskets and their neighboring parts have matched meshes, it is straightforward to connect gaskets to other components in a model simply by sharing nodes (see *Figure 3*).

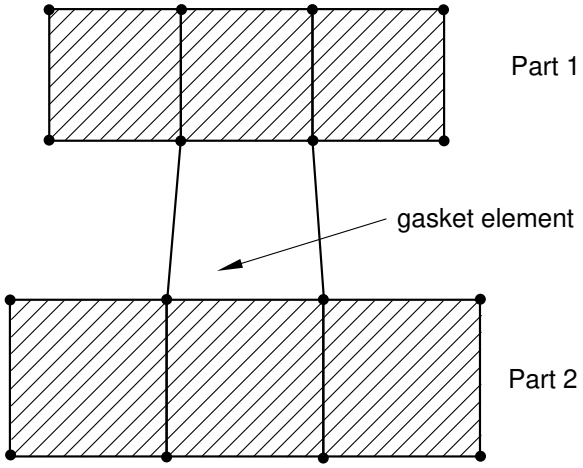


Figure 3: Gasket elements sharing nodes with other Abaqus elements.

This method of connecting gaskets to other components is suited for cases when no frictional slip occurs between the gasket and the other components. It can be used whether or not the membrane behavior of the gasket elements is defined; however, if the gasket membrane behavior is defined, using a contact pair approach will lead to more realistic results since the difference in membrane stiffness between the gasket and its neighboring parts may lead to frictional slip. The method of sharing nodes will also lead to some small tensile stresses in the gasket should the parts connected to the gasket be pulled apart, as a result of the numerical stabilization technique added to the gasket thickness-direction behavior (see *Defining the Gasket Behavior Directly Using a Gasket Behavior*

Model). The contact pair approach will avoid such tensile stresses. This node-sharing approach cannot be used with the gasket elements that consider only thickness-direction behavior.

Using Gasket Elements That Model Thickness-Direction Behavior Only

In general, the modeling techniques discussed earlier can be used with gasket elements that model thickness-direction behavior only. However, these elements have only one displacement degree of freedom per node and cannot share nodes with elements that have all displacement degrees of freedom active at a node. They can, however, share nodes with other gasket elements that model thickness-direction behavior only.

Discretizing a Gasket with Gasket Elements That Model Thickness-Direction Behavior Only

When discretizing a gasket with several layers of gasket elements along the gasket direction, it is recommended that all the nodes belonging to a cross-section of the gasket have the same thickness direction (see *Figure 4*). An approximate solution will be generated if the thickness direction changes, since only the magnitude of the force is transmitted from one gasket element to the next through the thickness of the gasket.

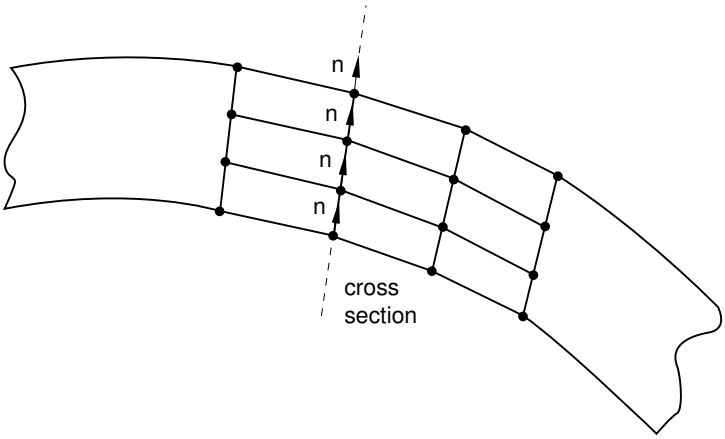


Figure 4: Discretizing a gasket using several layers of elements with thickness-direction behavior only.

Connecting Gaskets to Other Components When Gasket Elements with Thickness-Direction Behavior Only Are Chosen

Contact pairs can be used to connect the gasket mesh to adjacent components, as explained above, but only frictionless, small-sliding contact can be used.

MPC type PIN or TIE can also be used to connect a one degree of freedom node of a gasket element to another coincident node that has all its displacement degrees of freedom active (see *Figure 5*). Abaqus/Standard automatically constrains the single displacement degree of freedom node to the global displacements of the other node.

Surface-based tie constraints cannot be used to connect gasket elements that model thickness-direction behavior only.

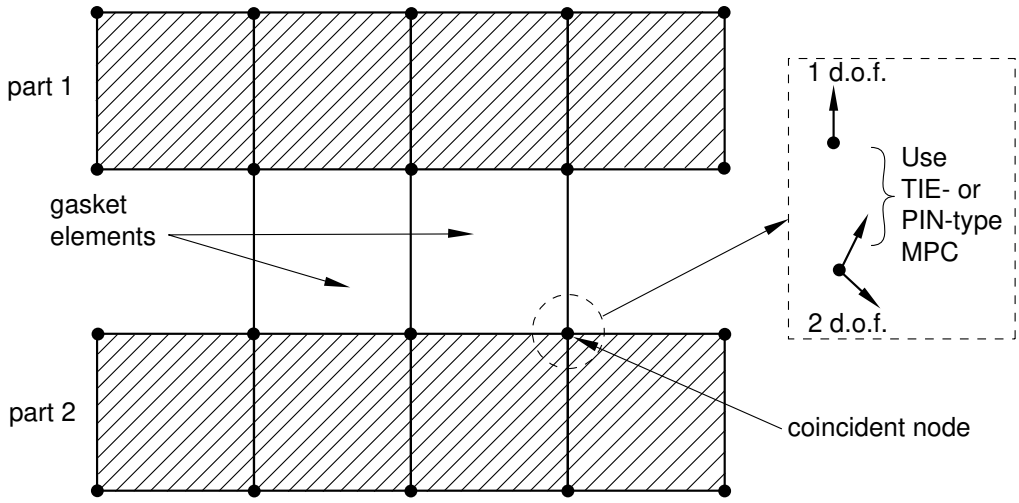


Figure 5: Connecting gasket elements with thickness-direction behavior only to other parts by using MPCs.

Additional Considerations When Using Gasket Elements

Several cases require special consideration when using gasket elements.

Using Gasket Elements in Large-Displacement Analyses

Gasket elements are small-strain, small-displacement elements. They can be used in large-displacement analyses. However, the local directions of the gasket elements are not updated with the solution, so incorrect results will be generated if the assembly containing the gasket elements undergoes any significant amount of rotation.

Using 12-Node Gasket Elements

These elements are primarily for use when the adjacent components are modeled with modified 10-node tetrahedral elements (element type C3D10M). When the contact pair approach is used, such elements can also be placed adjacent to other three-dimensional solid continuum elements; however, if the meshes are badly mismatched, the solution may be noisy.

Using 18-Node Gasket Elements

These elements are intended to share nodes with 21 to 27-node brick elements. They can also be connected to a mesh composed of 21 to 27-node brick elements or a mesh composed of 20-node brick elements when the contact pair approach is used.

Abaqus/Standard allows the node numbers and the coordinates of the midface nodes in the 18-node gasket elements to be generated automatically if the faces are part of contact surfaces, similar to the way that midface nodes are generated for 20-node brick element faces on which a contact surface is defined. This feature is invoked by leaving the entries for nodes 17 and 18 in the element connectivity blank.

Using the Three-Dimensional Line Gasket Elements

Three-dimensional line gasket elements are typically used to model narrow, thicker features in gaskets, such as an elastomeric insert around a hole. A typical mesh for such a case is presented in *Figure 6*. The gasket is discretized mainly with three-dimensional area elements. The insert is modeled with three-dimensional line elements that may or may not be connected to the area elements. These gasket elements are connected to surrounding components using two sets of contact pairs, and the area elements will typically have initial gaps

specified in the gasket property definition so that the thicker inserts develop pressure on contact before the area elements do.

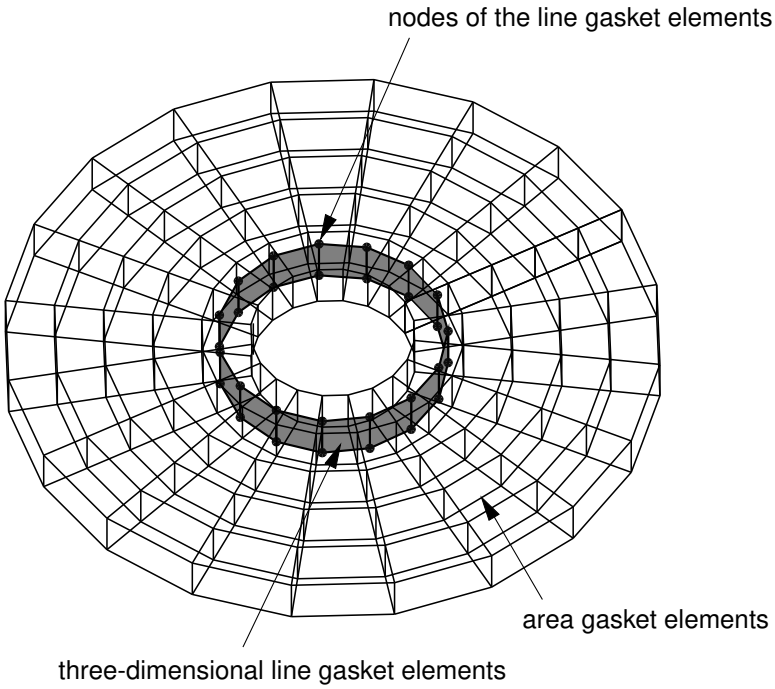


Figure 6: Typical use of three-dimensional line gasket elements to model inserts in gaskets.

If three-dimensional line gasket elements that have all displacement degrees of freedom active at their nodes are used to discretize a gasket and the local 3-direction is the same at all the nodes of these elements (this is the case when all elements lie in a plane), the nodes of these elements can move in the local 3-direction without creating any strain in the elements (see *Defining the Gasket Behavior Directly Using a Gasket Behavior Model* for additional details about the local direction of three-dimensional line elements). In such a case you should make sure that these elements are restrained properly in the local 3-direction.

Defining the Gasket Element's Initial Geometry

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Gasket Elements*
- **GASKET SECTION*
- *Creating gasket sections*

Overview

The initial gasket geometry:

- is defined by the nodal coordinates of the element; and
- is also defined by the thickness direction and initial thickness, each of which can be calculated by Abaqus/Standard or user-defined.

Defining the Element Geometry

A gasket element is basically composed of two surfaces (a bottom and a top surface) separated by the gasket thickness. The element has nodes on its bottom face and corresponding nodes on its top face.

Two methods are available to define the element geometry.

By Defining the Element's Nodes

You can define the geometry of the gasket element by defining the coordinates of all the element's nodes. You can define elements with constant or varying thickness. If the gasket element is very thin in comparison to dimensions in its surfaces, the thickness of the element calculated from the nodal coordinates may be inaccurate. In this case you can specify a constant thickness directly.

By Defining the Bottom Surface of the Element

You can specify a list of only the nodes on the bottom surface of the gasket element and the positive offset number that will be used to define the corresponding nodes on the top surface of the gasket element. Abaqus/Standard will create the nodes of the top face coincident with those of the bottom face unless the nodes of the top face have already been assigned coordinates. If the bottom and top nodes coincide, you must specify the thickness of the gasket element.

Specifying the Element Thickness

You can specify the gasket element thickness as part of its section property definition.

Input File Usage: **GASKET SECTION*
thickness

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Gasket** as the section **Type**: **Initial thickness: Specify:** *thickness*

Additional Quantities Needed to Specify the Element Geometry

For three-dimensional area elements, the element geometry is defined entirely by the location of the top and bottom surfaces and the element thickness. For two- and three-dimensional link elements (elements with two nodes, one on each face) you should specify the cross-sectional area of the element. For axisymmetric link elements you should specify the width of the element. For general two-dimensional elements the out-of-plane thickness is required. For three-dimensional line elements you should also specify the width of the element. This additional information is specified as part of the gasket section property definition; if it is not specified but is needed, it is assumed to have a value of 1.0.

Input File Usage: **GASKET SECTION*
, , , additional geometric data (cross-sectional area, width, or out-of-plane thickness)

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Gasket** as the section **Type**: **Cross-sectional area, width, or out-of-plane thickness:** *additional geometric data*

Default Element Thickness-Direction Definition

Gaskets are usually manufactured to have a desired behavior in their thickness direction. Therefore, it is important to define the thickness directions of gasket elements accurately. Abaqus/Standard computes these directions by default. The method that Abaqus/Standard uses depends on the gasket element type.

Link Elements

Abaqus/Standard computes the thickness direction for a two-dimensional, three-dimensional, or axisymmetric link element by subtracting the coordinates of node 1 from those of node 2, as shown in *Figure 1*. The computed thickness direction is then assigned to each node. If the gasket element is very thin, the thickness direction may not be predicted accurately. You can overwrite this direction, as explained below in *Specifying the Thickness Direction Explicitly*.

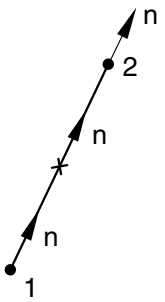


Figure 1: Thickness direction for a link element.

Two-Dimensional and Axisymmetric Elements

To compute the thickness direction for two-dimensional and axisymmetric elements, Abaqus/Standard forms a midsurface by averaging the coordinates of the node pairs forming the bottom and top surfaces of the element. This midsurface passes through the integration points of the element, as shown in *Figure 2*. For each integration point Abaqus/Standard computes a tangent whose direction is defined by the sequence of nodes given on the bottom and top surfaces. The thickness direction is then obtained as the cross product of the out-of-plane and tangent directions. The thickness direction computed at each integration point is then assigned to the nodes on either side of the integration point.

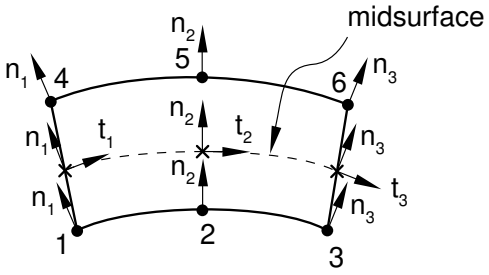


Figure 2: Thickness direction for a two-dimensional or axisymmetric element.

Three-Dimensional Area Elements

To compute the thickness direction for three-dimensional area elements, Abaqus/Standard forms a midsurface by averaging the coordinates of the node pairs forming the bottom and top surfaces of the element. This midsurface passes through the integration points of the element, as shown in *Figure 3*. Abaqus/Standard computes the thickness direction to the midsurface at each integration point; the positive direction is obtained with the right-hand rule going around the nodes of the element on the bottom or top surface. The thickness direction computed at each integration point is assigned to the nodes on either side of the integration point.

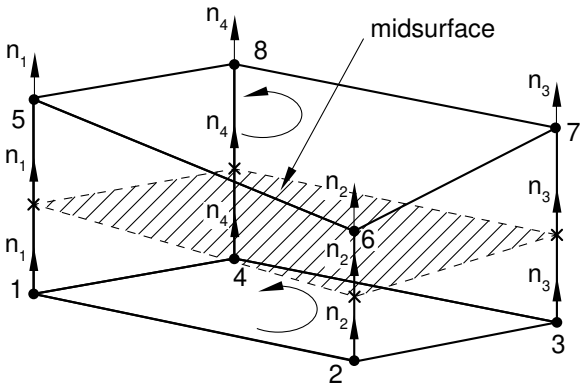


Figure 3: Thickness direction for a three-dimensional area element.

Three-Dimensional Line Elements

To compute the thickness direction for three-dimensional line elements, Abaqus/Standard computes the thickness direction at each integration point of the line element by differencing the coordinates of the element's surface nodes associated with the integration point. The thickness direction will point from the node on the bottom face to the node on the top face of the element. The thickness direction computed at each integration point is then assigned to the nodes on either side of the integration point (see *Figure 4*).

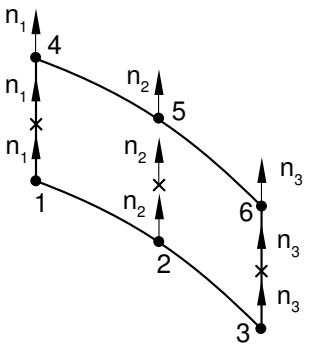


Figure 4: Thickness direction for a three-dimensional line element.

If the gasket element is very thin, the computation of the thickness direction may not be accurate. You can overwrite this definition as explained below in *Specifying the Thickness Direction Explicitly*.

Creating a Smooth Gasket

Gasket elements can be used in a single layer or can be stacked in multiple layers (see *Including Gasket Elements in a Model* for further details). The thickness directions computed at the nodes of gasket elements on an element-by-element basis are averaged at nodes shared by two or more gasket elements. This averaging process ensures that, if the gasket is not planar, it has a thickness direction that varies smoothly even though the gasket has been discretized by elements. You must ensure that the connectivities of the elements are such that the thickness direction does not reverse from one element to the next for this process to work properly. Once the averaging process is complete, the thickness directions at the nodes of a given element may vary significantly along the gasket midsurface and through its thickness, as shown in *Figure 5*. The thickness directions at any of the nodes of an element should not vary in direction by more than 20°. In addition, the thickness directions of two associated nodes through the thickness direction should not vary in direction by more than 5°. Abaqus/Standard will require that the gasket be remeshed when such conditions are not met.

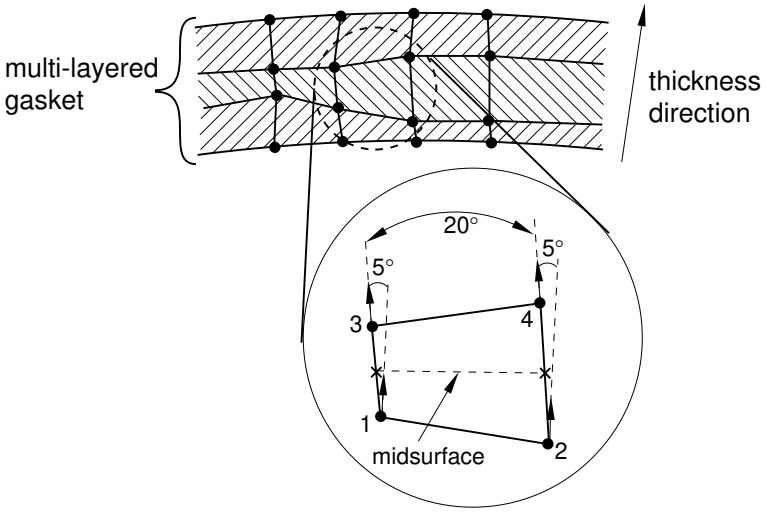


Figure 5: Result of the averaging process.

Specifying the Thickness Direction Explicitly

For cases when the above averaging process is not satisfactory, two methods are provided to specify the thickness direction of gasket elements.

Specifying the Thickness Direction as Part of the Gasket Section Definition

You can specify the components of the thickness direction as part of the gasket section definition. In this case all nodes of the gasket elements using this section definition are assigned the same thickness direction. The thickness direction specified at the nodes of the element will be averaged at nodes shared by two or more elements.

Input File Usage: **GASKET SECTION*
, , , , component 1, component 2, component 3

Abaqus/CAE Usage: You cannot specify the gasket thickness direction in Abaqus/CAE.

Specifying the Thickness Direction by Specifying a Normal Direction at the Nodes

You can define the thickness direction at a particular integration point of a gasket element by specifying a normal direction for the node on the bottom face of the element that is associated with the integration point (see *Normal Definitions at Nodes*). The thickness direction will not be averaged if this node belongs to more than one element. The thickness direction specified at the bottom node will also be assigned at the top node associated with the same integration point. This thickness direction will not be averaged if the top node belongs to more than one element; however, you can overwrite this thickness direction by specifying a normal at this node if it is the bottom node of another element. This last situation can occur only in cases when gasket elements are stacked up through the thickness direction of the gasket. If this method is used to specify conflicting thickness directions at the same node, Abaqus/Standard will issue an error message. Thickness directions specified using this method will overwrite any thickness directions specified at a gasket node as part of the gasket section definition.

Input File Usage: **NORMAL*

Abaqus/CAE Usage: User-specified nodal normals are not supported in Abaqus/CAE.

Creating Fold Lines

It is possible to introduce a fold line in a gasket by creating gaskets with coincident nodes and using MPC type TIE or PIN (*General Multi-Point Constraints*) to constrain the displacement of these nodes. However, fold lines are rarely needed in the analysis of gaskets, since almost all gaskets are manufactured with smoothly varying surfaces.

Verifying the Thickness Direction

Thickness direction definitions can be checked by examining the analysis input file processor output. The direction cosines of the thickness directions obtained at the nodes of gasket elements are listed under `GASKET THICKNESS DIRECTIONS` in the data (`.dat`) file.

Specifying an Initial Gap and an Initial Void in the Thickness Direction of a Gasket Element

The construction of gaskets in their through-thickness direction may be complex; for example, certain automotive gaskets are usually composed of several layers of metal and/or elastomeric inserts, and it is likely that the layers do not all touch until the gasket is compressed. The inter-layer spaces in a gasket are referred to in Abaqus as the initial void. The initial void is used only for calculating thermal strain and creep strain. It is also possible that the gasket surface geometry is such that pressure will not start building up until the gasket has been compressed by a certain amount. The gasket closure that is needed to generate a pressure is referred to in Abaqus as the initial gap. *Figure 6* shows a schematic representation of the initial gap and initial void in a typical gasket. You can specify both the initial gap and initial void as part of the gasket section property definition. The initial thickness of the element should include the initial gap and the initial void.

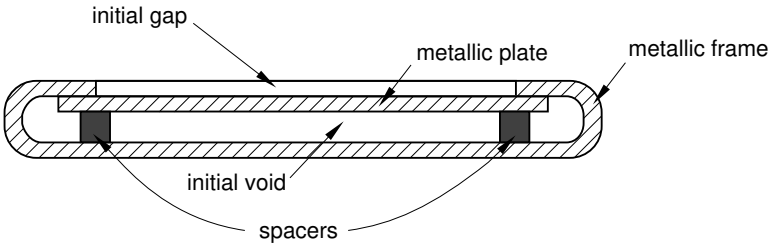


Figure 6: Schematic representation of an initial gap and an initial void in a typical gasket.

Input File Usage: **GASKET SECTION*
, *initial gap*, *initial void*

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Gasket** as the section **Type**: **Initial gap:** *initial gap*, **Initial void:** *initial void*

Stability of Unsupported Gasket Elements

Gasket elements that extend outside neighboring components (unsupported gasket elements) can be troublesome and should be avoided. If a gasket element is completely or partially unsupported, incorrect areas can result in an incorrect stiffness, and numerical singularity problems can occur in the equation solver. Minor extensions (caused by numerical roundoff in mesh generation) will not usually cause a problem because Abaqus/Standard automatically extends the main surfaces a small amount beyond the edge of the model. Numerical problems can occur in the direction tangential to the gasket (if general gasket elements are used and no membrane stiffness is specified) as well as in the direction normal to the gasket. The numerical singularity problems normal to the gasket can be treated by stabilizing the elements with a small artificial stiffness. By default, Abaqus/Standard automatically applies a small stabilization stiffness (on the order of 10^{-9} times the initial compressive stiffness in the thickness direction) to all types of gasket elements except the link elements. For persistent numerical singularity problems in unsupported gasket elements the following treatment methods can be considered. First, make sure that an adequate membrane elasticity is specified. Second, specify a higher value for the artificial stiffness for the gasket section. If problems still persist, consider trimming, "skinning," and using MPCs (see *General Multi-Point Constraints*).

Input File Usage: Use the following option to change the artificial stiffness for a gasket section:

**GASKET SECTION*, STABILIZATION STIFFNESS=*stiffness_value*

Abaqus/CAE Usage: Use the following option to change the artificial stiffness for a gasket section:

Property module: **Create Section**: select **Other** as the section **Category** and **Gasket** as the section **Type**: **Stabilization stiffness: Specify:** *stiffness_value*

Defining the Gasket Behavior Using a Material Model

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Gasket Elements*
- *UMAT*
- *Creating and editing materials*

Overview

The gasket behavior defined by a material model:

- can be specified in terms of a built-in material model or a user-defined small-strain material model;
- considers only the thickness behavior and assumes a uniaxial stress state for gasket elements that model thickness-direction behavior only;
- admits both compressive and tensile stresses in the thickness direction;
- is defined in terms of small-strain measures and, hence, finite-strain material models such as hyperelastic and hyperfoam cannot be used;
- is restricted to small-strain elasticity models for line gasket elements that use the built-in material models;
- causes Abaqus/Standard to use the reference thickness to convert the relative displacements at the top and bottom surfaces of the gasket to strains and uses these strains in conjunction with the constitutive law to obtain the stresses; and
- makes the notions of “initial gap” and “initial void” in the thickness direction irrelevant (consequently, Abaqus/Standard ignores such data specified as part of the gasket section property definition).

Assigning a Gasket Behavior to a Gasket Element

To define the gasket behavior by a material model, you must assign a gasket section definition to a region of your model and assign the name of a material definition to the gasket section definition. The gasket behavior for this region is defined entirely by the gasket thickness and the material properties specified by the material definition referring to the same name.

The gasket behavior can be defined in terms of a built-in or a user-defined material model. In the latter case the actual material model is defined in user subroutine *UMAT*.

Input File Usage: Use the following options to define the gasket behavior in terms of a built-in material model:

**GASKET SECTION*, ELSET=*name*, MATERIAL=*name*
**MATERIAL*, NAME=*name*

Use the following options to define the gasket behavior in terms of a user-defined material model:

**GASKET SECTION*, ELSET=*name*, MATERIAL=*name*
**MATERIAL*, NAME=*name*
**USER MATERIAL*, CONSTANTS=*n*

Abaqus/CAE Usage: Property module:
Create Material: **Name:** *name*, enter data for any materials that are valid for gasket sections except those found under **Other->Gasket**
Create Section: select **Other** as the section **Category** and **Gasket** as the section **Type**: **Material:** *name*

Tensile Behavior Modeling

Tensile behavior modeling can be desirable when gaskets carry (limited) tensile stresses, such as occurs when adhesives are present. Undesired tensile behavior can be avoided by using appropriate contact pairs and/or implementing a user-defined no-tension material model in user subroutine *UMAT*.

Specific Output for Material Definition of Gasket Behavior

The output variables for stresses and strains are the same as those used for solid elements: tensile and compressive stresses/strains are indicated as positive and negative quantities, respectively. However, for all stress/strain output variables the 11-component refers to the through-thickness direction; the 22-, 33-, and 23-components refer to two direct and one shear membrane component, respectively; the remaining 12- and 13-components refer to the transverse shear components. For details about these definitions, see *About Gasket Elements*. The output variable NE is available to output nominal (effective) strains for gasket elements defined using a material model; however, NE is identical to E in this case.

Defining the Gasket Behavior Directly Using a Gasket Behavior Model

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Gasket Elements*
- *Defining the Gasket Element's Initial Geometry*
- *Defining gasket behavior*

Overview

The gasket behavior defined by a gasket behavior model:

- can be specified in terms of uncoupled thickness direction, membrane, and transverse shear behavior only;
- can be nonlinear elastic with damage or nonlinear elastic-plastic in the thickness direction;
- can consider creep effects in the thickness direction when rate-independent elastic-plastic modeling is used;
- can consider the dynamic stiffness and damping characteristics in the thickness direction when elastic-damage modeling is used;
- is linear elastic in the membrane and transverse shear directions; and
- can consider thermal effects in the thickness and membrane directions.

Assigning a Gasket Behavior to a Gasket Element

To define the gasket behavior by a gasket behavior model, you must assign a gasket section definition to a region of your model and assign the name of a gasket behavior definition to the gasket section definition. The gasket behavior for this region is defined entirely by the properties specified by the gasket behavior definition referring to the same name.

Input File Usage: Use both of the following options to define the gasket behavior in terms of a gasket behavior model:

**GASKET SECTION*, ELSET=*name*, BEHAVIOR=*name*
**GASKET BEHAVIOR*, NAME=*name*

Abaqus/CAE Usage: Property module:
Material editor: **Name:** *name*, enter data for any materials found under **Other->Gasket**
Create Section: select **Other** as the section **Category** and **Gasket** as the section **Type**: **Material:** *name*

Specifying a Gasket Behavior

The thickness-direction, transverse shear, and membrane behaviors are defined to be uncoupled. Each behavior is specified independently.

You must specify the thickness-direction behavior. You can specify multiple thickness-direction behaviors to define the loading and unloading characteristics. You can obtain an average contact pressure output when the thickness-direction behavior is defined as force or force per unit length versus closure.

The transverse shear and membrane behaviors are optional for gasket elements that have all displacement degrees of freedom active at their nodes. You can define one or both of these behaviors.

When thermal and rate-dependent effects are important, you can define thermal expansion and creep behavior for gaskets; user subroutines *UEXPAN* and *CREEP* can be used to define these behaviors.

You cannot specify density for gasket elements since they have no mass matrix.

Input File Usage: Use the first two options and any of the following options to specify a gasket behavior:

**GASKET BEHAVIOR*, NAME=*name*
**GASKET THICKNESS BEHAVIOR*
**GASKET ELASTICITY*
**GASKET CONTACT AREA*
**EXPANSION*
**CREEP*
**DEPVAR*
**USER OUTPUT VARIABLES*

The **GASKET THICKNESS BEHAVIOR* option can be repeated to define the loading and unloading characteristics of the thickness-direction behavior. The **GASKET ELASTICITY* option can be repeated to define both transverse shear and membrane behaviors. The other options cannot be repeated within a single behavior definition. The order in which these options are specified has no importance, but they must appear immediately after the **GASKET BEHAVIOR* option.

Abaqus/CAE Usage: Use the first option and any of the following options to specify a gasket behavior:

Property module: material editor:
Other->Gasket->Gasket Thickness Behavior
Other->Gasket->Gasket Transverse Shear Elasticity and/or **Gasket Membrane Elasticity**
Mechanical->Expansion
Mechanical->Plasticity->Creep
General->Depvar
General->User Output Variables

Defining the Thickness-Direction Behavior of the Gasket

To define the thickness-direction behavior of gaskets, Abaqus/Standard offers a nonlinear elastic model with damage and a nonlinear elastic-plastic model with the possibility of considering creep effects. Thermal effects in the thickness direction can also be accounted for.

Abaqus/Standard measures the thickness-direction deformation as the closure between the bottom and top faces of the gasket element; therefore, the thickness-direction behavior must always be defined in terms of closure. The closure is the sum of the elastic closure, plastic closure, creep closure, thermal closure, plus any initial gap in the thickness direction. As explained below, the behavior can be defined as pressure versus closure, force versus closure, or force per unit length versus closure. In all cases the thickness-direction behavior can be defined as a function of temperature and/or field variables.

Input File Usage: **GASKET THICKNESS BEHAVIOR*, DEPENDENCIES

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Thickness Behavior**

Choosing a Unit System Used to Define the Thickness-Direction Behavior

The thickness-direction behavior can be defined in terms of pressure versus closure, force versus closure, or force per unit length versus closure.

Prescribing the Thickness-Direction Behavior as Pressure Versus Closure

You can define the thickness-direction behavior in terms of pressure and closure for all gasket element types. The pressure is available for output or visualization.

Input File Usage: **GASKET THICKNESS BEHAVIOR*, VARIABLE=STRESS

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Thickness Behavior**: **Units: Stress**

Prescribing the Thickness-Direction Behavior as Force or Force per Unit Length Versus Closure

You can define the thickness-direction behavior in terms of force or force per unit length and closure only for link elements and three-dimensional line elements. This method is suited for cases where the gasket cross-section in the 1–2 or 1–3 plane varies greatly with deformation because it would be too expensive to model such a deformation with a full two- or three-dimensional model. In such cases a model with link elements or three-dimensional line elements can give meaningful answers if the deformation is quantified in terms of force or force per unit length (see *Figure 1*).

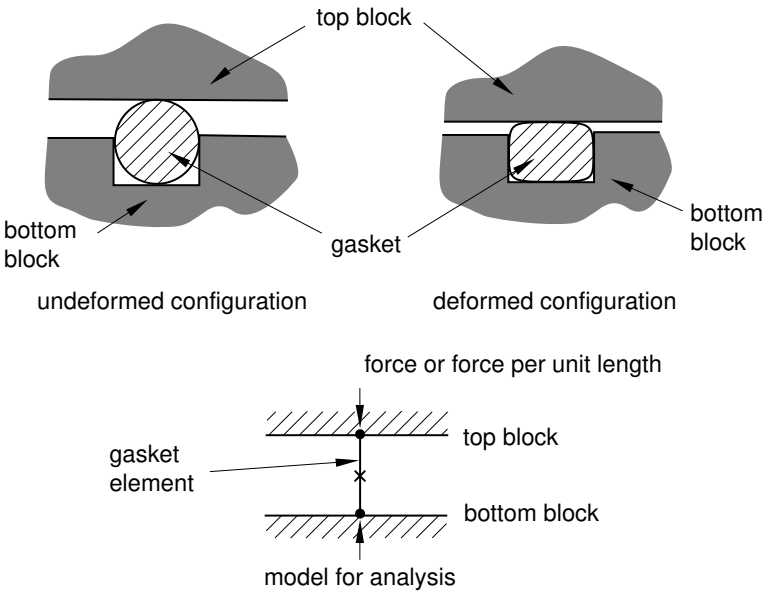


Figure 1: Modeling complex deformations with link or three-dimensional line elements.

When using two- or three-dimensional link elements, you must specify the thickness-direction behavior as force versus closure. When using axisymmetric link elements or three-dimensional line elements, you must specify the thickness-direction behavior as force per unit length versus closure.

Input File Usage: **GASKET THICKNESS BEHAVIOR*, VARIABLE=FORCE

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Thickness Behavior**: **Units: Force**

Defining a Nonlinear Elastic Model with Damage

The nonlinear elastic model with damage is illustrated in *Figure 2*.

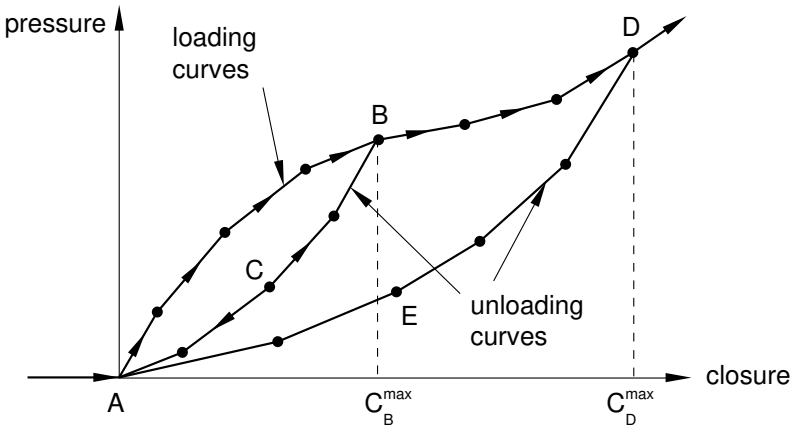


Figure 2: Elastic model with damage.

As the gasket is compressed, the pressure (or force, or force per unit length) follows the path given by the loading curve. If the gasket is unloaded, for example at point *B*, the pressure follows the unloading curve BCA. Reloading after unloading follows the unloading curve ACB until the loading is such that the closure becomes greater than C_B^{max}, after which the loading path follows the loading curve BD. The arrows shown in the figure illustrate the loading/unloading paths of this model.

Defining the Loading Curve

To define the loading curve in piecewise linear form, you provide data points of pressure versus elastic closure, starting with point *A*. For negative elastic closures, the model gives zero pressure (or force). For closures larger than the last user-specified closure, the pressure-closure relationship is extrapolated based on the last slope computed from the user-specified data.

Input File Usage: **GASKET THICKNESS BEHAVIOR*, TYPE=DAMAGE, DIRECTION=LOADING

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Thickness Behavior**: **Type: Damage**, **Loading**

Defining the Unloading Curve

To define the unloading curves (ACB, AED, and so on), you provide data points of pressure (or force) versus elastic closure up to a given maximum closure (C_B^{max}, or C_D^{max}, and so on). You can specify as many unloading curves as are necessary. Each unloading curve always starts at point *A*, the point of zero pressure for zero elastic closure, since the damaged elasticity model prevents any permanent deformation. If unloading occurs from a maximum closure for which an unloading curve is not specified, the unloading is interpolated from neighboring unloading curves. The unloading curves are stored in normalized form so that they intersect the loading curve at a unit stress (or unit force) for a unit elastic closure, and the interpolation occurs between these normalized curves.

This approach works very well in most situations. However, there are situations where interpolation based on normalized curves can lead to some inaccuracies. One example of such a situation is if the unloading occurs from a maximum closure that is very close to but smaller than the maximum closure of the first specified unloading curve (point *B* in *Figure 2*). Another example of such a situation occurs at relatively high values of closure where

the loading curve of the gasket behavior shoots up and becomes steep, causing the last specified unloading curve to be relatively close to the loading curve.

To overcome these inaccuracies, Abaqus offers an alternative interpolation approach that is based directly on the specified curves. In this approach the specified curves are used directly, and no normalization is performed before the interpolation. The alternative interpolation scheme is illustrated in *Figure 3*. Assume that you have specified a loading curve and two unloading curves at maximum closures of C_D^{max} and C_B^{max}, respectively. During unloading, the pressure at a given closure is interpolated from two of the specified curves, one of which may be the loading curve. For example, if unloading occurs from point *F* on the loading curve in *Figure 3*, for all values of closure greater than C_B^{max}, the pressure is interpolated based on the loading curve and the unloading curve corresponding to the maximum closure of C_D^{max}. Therefore, the pressure at point *I* on the unloading curve is computed as:

$$p_I = \frac{(C_F - C_G)\, p_K + (C_D^{max} - C_F)\, p_G}{C_D^{max} - C_G}, \text{if } C_B^{max} < C_G < C_D^{max}.$$

However, for all values of closure smaller than C_B^{max}, the pressure is interpolated based on the two unloading curves corresponding to the maximum closures of C_D^{max} and C_B^{max}, respectively. Therefore, the pressure at point *J* is computed as:

$$p_J = \frac{(C_F - C_B^{max})\, p_L + (C_D^{max} - C_F)\, p_H}{C_D^{max} - C_B^{max}}, \text{if } C_J < C_B^{max}.$$

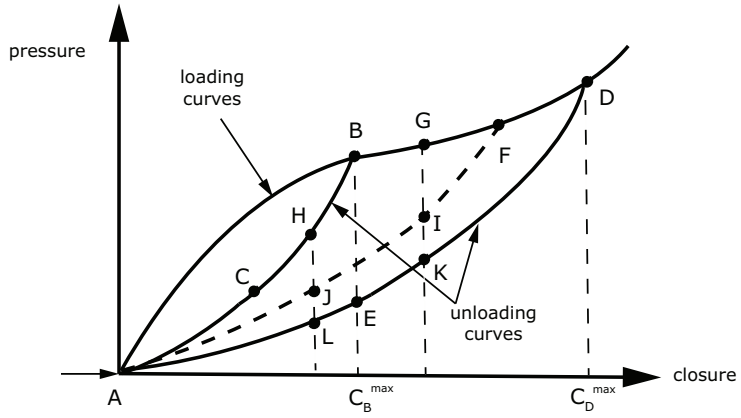


Figure 3: Alternative interpolation scheme for unloading curves.

If unloading curves are not specified, the loading/unloading follows the loading curve.

Input File Usage: Use the following option to use the interpolation scheme based on the normalized curves:

**GASKET THICKNESS BEHAVIOR*, TYPE=DAMAGE,
DIRECTION=UNLOADING, INTERPOLATION=NORMALIZED (default)

Use the following option to use the alternative interpolation scheme:

**GASKET THICKNESS BEHAVIOR*, TYPE=DAMAGE,
DIRECTION=UNLOADING, INTERPOLATION=DIRECT

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Thickness Behavior**: **Type: Damage**, **Unloading**, toggle on **Include user-specified unloading curves**

Defining unloading curves using the alternative interpolation scheme is not supported in Abaqus/CAE.

Defining the Behavior for Elements with an Initial Gap

For cases when the load in the gasket does not increase as soon as the gasket is compressed (see *Figure 4*), you can specify an initial gap as part of the gasket section property definition (see *Defining the Gasket Element's Initial Geometry*) and define the loading/unloading curves as if the initial gap were not present (the case of *Figure 2*). This method is convenient when many gasket elements refer to the same gasket behavior and the only difference is the initial gap.

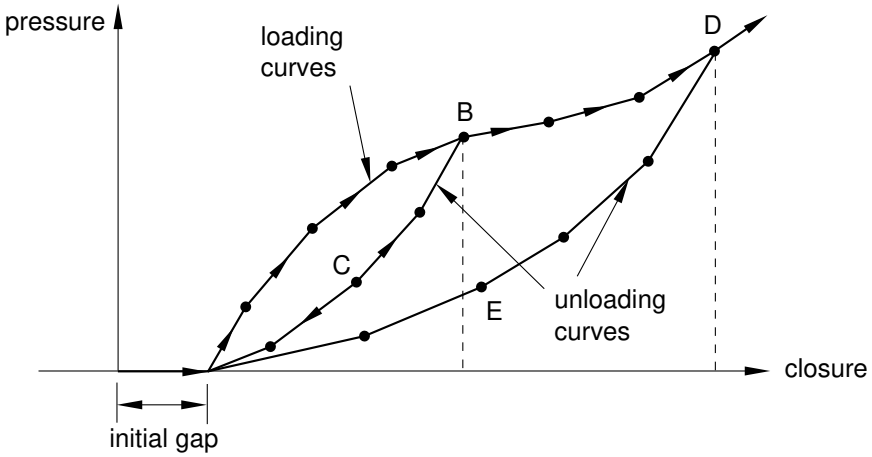


Figure 4: Elastic model with damage and initial gap.

Defining a Nonlinear Elastic-Plastic Model

The nonlinear elastic-plastic model is illustrated in *Figure 5*.

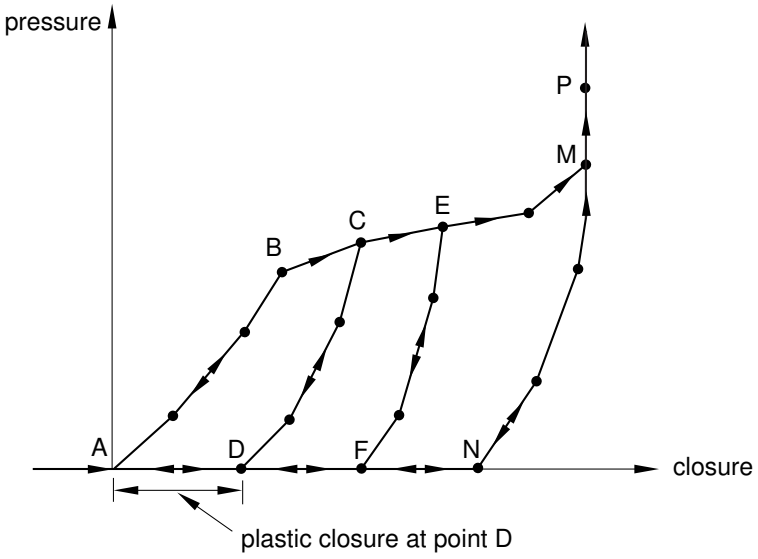


Figure 5: Elastic-plastic model.

As the gasket is compressed, the pressure (or force) follows the path given by the loading curve $ABCEM$. The loading curve is a nonlinear elastic curve until point *B* is reached. At point *B* the slope of the loading curves decreases by more than 10%, which is assumed to correspond with the onset of plastic deformation. The value of 10% was chosen as a reasonable minimum value that can be expected at the onset of yield. If yield starts at a point at which no decrease in the slope occurs, numerical difficulties may occur. If the elastic part of the loading curve has a changing slope, the curve should be defined such that the slope does not decrease by more than 10% at any given point. After point *B* plastic deformation starts taking place. If unloading occurs before point *B* is reached, unloading takes place along the initial loading curve. Once loading has gone beyond point *B*, unloading takes place along an unloading curve such as curve CD. The unloading is assumed to be entirely elastic. The amount of closure at point *D* represents the plastic closure for the unloading curve CD. Reloading after unloading follows the same curve DC until the gasket yields, after which the loading curve CEM is followed. Plastic deformation takes place until the last point *M* on the loading curve is reached. Beyond point *M*, the curve NP is followed for both loading and unloading; this behavior represents the behavior of a crushed gasket, which is assumed to be entirely elastic and can be specified in a piecewise-linear fashion, even beyond point *M*. The arrows shown in the figure illustrate the loading/unloading paths for the elastic-plastic model.

Abaqus/Standard automatically converts the curves so that the unloading curves become curves of pressure (or force) versus elastic closure for a given plastic closure. The loading curve is transformed into an elastic loading/unloading curve defined at zero plastic closure (the portion AB of the curve) and a yield curve (the portion BM of the curve). By default, the onset of yield (point *B*) is obtained as soon as the slope of the loading curve decreases by 10% from the maximum slope recorded up to that point while traveling along the loading curve from point *A* to point *M*. Abaqus/Standard offers two alternatives to allow you to override this default method of determining the onset of yield as described below. If only a loading curve is provided, the unloading is based on the curve AB, independent of the level of plasticity.

Defining the Loading Curve

To define the loading curve in piecewise linear form, you provide data points of pressure (or force, or force per unit length) versus closure (where closure represents the elastic plus the plastic closure), starting with point *A*. The last closure value given represents the closure at which the gasket is assumed crushed (point *M* in *Figure 5*); at this point, the maximum permanent deformation is reached. For negative closures the model gives zero pressure (or force).

To override the default method of determining the onset of yield, you can specify either a value for the decrease in slope other than the default value of 10% or the closure value at which onset of yield occurs. The specified value must correspond to a point on the loading curve at which the slope decreases.

Input File Usage: Use the following option to define the loading curve and use the default method for determining the onset of yield:

**GASKET THICKNESS BEHAVIOR*, TYPE=ELASTIC-PLASTIC, DIRECTION=LOADING

Use the following option to define the loading curve and specify a nondefault value for the decrease in slope that defines the onset of yield:

**GASKET THICKNESS BEHAVIOR*, TYPE=ELASTIC-PLASTIC, DIRECTION=LOADING, SLOPE DROP=*drop*

Use the following option to define the loading curve and specify the closure value that defines the onset of yield:

**GASKET THICKNESS BEHAVIOR*, TYPE=ELASTIC-PLASTIC, DIRECTION=LOADING, YIELD ONSET=*closure_value*

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Thickness Behavior**: **Type: Elastic-Plastic**, **Loading**, **Yield onset method: Relative slope drop** *drop* or **Yield onset method: Closure value** *closure_value*

Defining the Unloading Curve

To define the unloading curves (***CD***, ***EF***, and so on), you provide data points of pressure (or force, or force per unit length) versus closure (elastic plus plastic) for each given plastic closure (closure at points *D*, *F*, and so on) in ascending values of closure. You can specify as many unloading curves as are necessary. If unloading occurs at a plastic closure for which an unloading curve is not specified, the unloading curve is interpolated from neighboring unloading curves. If no unloading curves are specified, unloading is assumed to follow a curve similar to the initial nonlinear elastic segment of the loading curve. The unloading curves are stored in normalized form so that they intersect the yield curve at a unit stress (or unit force) for a unit elastic closure, and the interpolation occurs between these normalized curves.

If the loading curve includes highly nonlinear behavior after the onset of yield, the interpolated unloading may give unreasonable behavior (such as the interpolated unloading path crossing over the user-defined loading curve). You should specify as many user-defined unloading curves as are required to create regions for which interpolated unloading response is appropriate. For example, *Figure 6* illustrates a loading curve that includes a sharp decrease in the hardening slope well after the onset of yield.

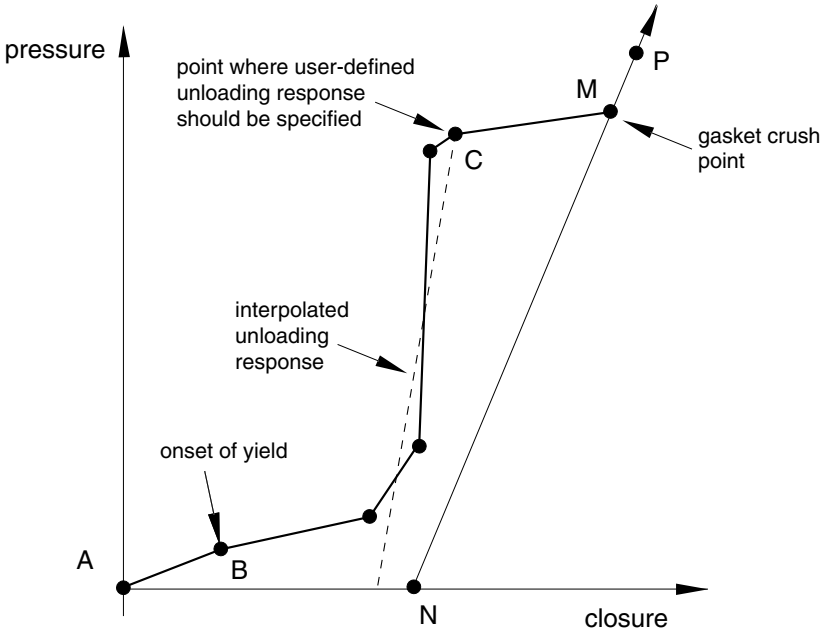


Figure 6: Elastic-plastic behavior with complex loading curve.

In this case it is insufficient to specify only one unloading curve at the gasket crush point (the end of the loading data). If unloading were to take place from point *C*, the unloading path would cross over the loading path. At least one additional unloading curve is required, after the sharp decrease in hardening slope, to prevent the interpolated unloading path crossing the loading curve.

Input File Usage: **GASKET THICKNESS BEHAVIOR*, TYPE=ELASTIC-PLASTIC, DIRECTION=UNLOADING

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Thickness Behavior**: **Type: Elastic-Plastic**, **Unloading**, toggle on **Include user-specified unloading curves**

Defining the Behavior for Elements with an Initial Gap

For cases when the load in the gasket does not increase as soon as the gasket is compressed (see *Figure 7*), you can specify an initial gap as part of the gasket section property definition (see *Defining the Gasket Element's Initial Geometry*) and define the loading/unloading curves as if the initial gap were not present (the case of *Figure 5*). This method is convenient when many gasket elements refer to the same gasket behavior and the only difference is the initial gap.

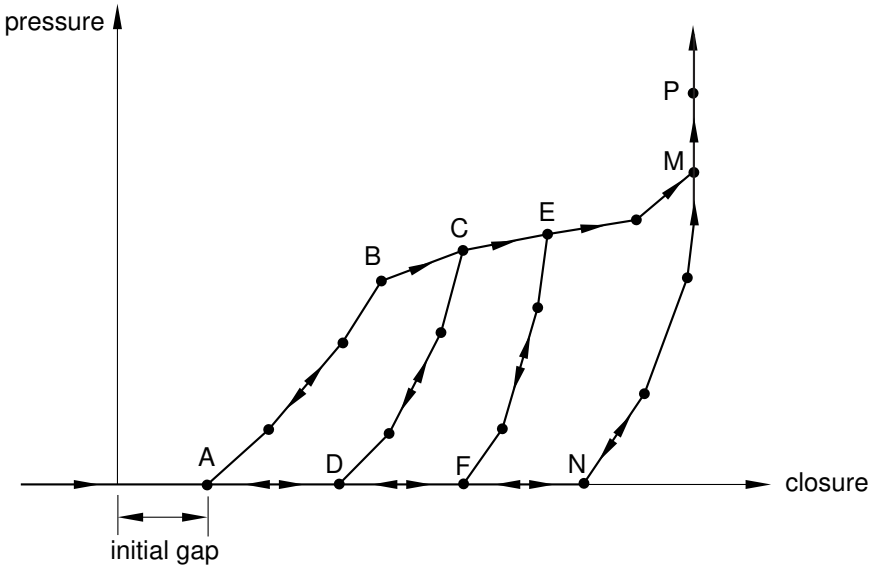


Figure 7: Elastic-plastic model with initial gap.

Numerical Stabilization of the Thickness-Direction Behavior

The damage and elastic-plastic models described above have zero stiffness at zero pressure. To overcome numerical problems caused by this zero stiffness, Abaqus/Standard automatically adds a small stiffness (by default, equal to 10^{-3} times the initial compressive stiffness) in the thickness direction of the gasket when the pressure obtained from the specified gasket thickness behavior is zero. This numerical stabilization ensures that the gasket element always returns to its stress-free thickness when it is totally unloaded. Hence, if the gasket surfaces are pulled apart, a small force arises from the stabilization process. You can change the default stiffness.

Input File Usage: **GASKET THICKNESS BEHAVIOR*, DIRECTION=LOADING,
TENSILE STIFFNESS FACTOR=*factor*

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Thickness Behavior**: **Loading**, **Tensile stiffness factor:** *factor*

Defining the Transverse Shear Behavior of the Gasket

You can define the elastic transverse shear stiffness of the gasket. Abaqus/Standard measures the relative displacement between the bottom and top of the gasket element along the local 2- or 3-directions to define the transverse shear in the gasket. Therefore, you should always define the elastic transverse stiffness as stress (or

force, or force per unit length) per unit displacement. You can specify the stiffness as a function of temperature and field variables. The same stiffness is used for the shear in the 1–2 plane and the shear in the 1–3 plane. For each set of temperature and/or field variables, the first slope of the initial loading curve for the gasket's thickness-direction behavior is used to compute the transverse shear stiffness if the transverse shear behavior is not defined explicitly.

Input File Usage: **GASKET ELASTICITY*, COMPONENT=TRANSVERSE SHEAR, DEPENDENCIES

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Transverse Shear Elasticity**

Choosing a Unit System to Define the Transverse Shear Behavior

The transverse shear stiffness is defined with units of stress per unit displacement, force per unit displacement, or force per unit length per unit displacement. The unit system used to define the transverse shear behavior must be consistent with the unit system used for the thickness-direction behavior.

Providing the Stiffness with Units of Stress per Unit Displacement

You can define the transverse shear stiffness in units of stress per unit displacement for all gasket element types. The stiffness is used to compute transverse shear stresses, which are available for output or visualization.

Input File Usage: **GASKET ELASTICITY*, COMPONENT=TRANSVERSE SHEAR, VARIABLE=STRESS

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Transverse Shear Elasticity**: **Units: Stress**

Providing the Stiffness with Other Units

You can define the transverse shear stiffness in units of force (or force per unit length) per unit displacement only for link elements and three-dimensional line elements. This method is suited for cases where the gasket cross-section in the 1–2 or 1–3 plane varies greatly with deformation because it would be too expensive to model such a deformation mechanism with a full two- or three-dimensional model, as explained earlier.

When using two- or three-dimensional link elements, you must specify the stiffness in terms of units of force per unit displacement. Abaqus/Standard uses this stiffness to compute transverse shear forces, which are available for output or visualization. When using axisymmetric link elements and three-dimensional line elements, you must specify the stiffness in terms of units of force per unit length per unit displacement. Abaqus/Standard uses this stiffness to compute transverse shear forces per unit length, which are available for output or visualization.

Input File Usage: **GASKET ELASTICITY*, COMPONENT=TRANSVERSE SHEAR, VARIABLE=FORCE

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Transverse Shear Elasticity**: **Units: Force**

Defining the Membrane Behavior of the Gasket

You can define the linear elastic behavior of the gasket by giving Young's modulus and Poisson's ratio. These data can be provided as a function of temperature and/or field variables. If you do not specify the linear elastic behavior of the gasket, the gasket has no membrane stiffness. In this case you must ensure that the nodes of the elements are restrained adequately in the directions orthogonal to the thickness direction of the gasket.

Input File Usage: **GASKET ELASTICITY*, COMPONENT=MEMBRANE, DEPENDENCIES

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Membrane Elasticity**

Defining Thermal Expansion for the Membrane and Thickness-Direction Behaviors

You can define isotropic thermal expansion to specify the same coefficient of thermal expansion for the membrane and thickness-direction behaviors.

Alternatively, you can define orthotropic thermal expansion to specify three different coefficients of thermal expansion. The first coefficient applies to the thermal expansion of the gasket in the thickness direction; the other two coefficients apply to the expansion of the gasket in the local 2- and 3-directions, respectively.

The membrane thermal strains, ε^{th}, are obtained as explained in *Thermal Expansion*. Abaqus/Standard computes the thermal closure for the thickness direction as

$$C^{th} = \varepsilon^{th} * (\text{initial gap} + \text{initial void} - \text{initial thickness}),$$

so that the "mechanical" closure is obtained as

$$C^{mech} = C^{total} - C^{th}.$$

You can specify the initial gap and initial void as part of the gasket section definition; the initial thickness is obtained directly from the nodal coordinates of the gasket elements, or you can specify it as part of the gasket section definition (see *Defining the Gasket Element's Initial Geometry*).

If user subroutine *UEXPAN* is used to define the thermal expansion of the gasket, the incremental thermal strains must be provided in the subroutine. The thermal closure is obtained from the thermal strain in the thickness direction, as described above.

Input File Usage: Use either of the following options to define the thermal expansion directly:

**EXPANSION*, TYPE=ISO
**EXPANSION*, TYPE=ORTHO

Use either of the following options to define the thermal expansion in user subroutine *UEXPAN*:

**EXPANSION*, TYPE=ISO, USER
**EXPANSION*, TYPE=ORTHO, USER

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Expansion**: **Use user subroutine UEXPAN** (optional)

Defining Creep Behavior for the Thickness-Direction Behavior

You can define creep behavior in the thickness direction of the gasket only when the elastic-plastic model (see *Defining a Nonlinear Elastic-Plastic Model* above) is used. The creep closure rate is obtained as

$$\dot{C}^{cr} = \dot{\varepsilon}^{cr} * (\text{initial thickness} - \text{initial gap} - \text{initial void}),$$

where $\dot{\varepsilon}^{cr}$ is obtained as explained in *Rate-Dependent Plasticity: Creep and Swelling*. You can specify the initial gap and initial void as part of the gasket section definition; the initial thickness is obtained directly from the nodal coordinates of the gasket elements, or you can specify it as part of the gasket section definition (see *Defining the Gasket Element's Initial Geometry*).

If user subroutine *CREEP* is used to define the rate-dependent thickness-direction response of the gasket, the compressive creep strain increment must be provided in the subroutine. The creep closure is obtained from the creep strain, as described above.

Input File Usage: Use the following option to define the creep behavior directly:

**CREEP*

Use the following option to define the creep behavior in user subroutine *CREEP*:

**CREEP*, LAW=USER

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Plasticity->Creep**: **Law: User-defined** (optional)

Defining Viscoelastic Behavior for the Thickness-Direction Behavior

You can define viscoelastic behavior in the thickness direction of the gasket only when the elastic-damage model (see *Defining a Nonlinear Elastic Model with Damage* above) is used. Only frequency domain viscoelastic behavior is supported. This behavior is useful for modeling the steady-state dynamic response of automotive components with gaskets about some pre-loaded base state, such as would be obtained at the end of a nonlinear sealing analysis, to determine the noise-vibration-harshness (NVH) characteristics of the system.

During the nonlinear sealing analysis step the frequency-domain viscoelastic behavior is ignored, and the material response is determined by the long-term elastic properties of the material. It is generally accepted (Zubeck and Marlow, 2002) that the dynamic stiffness and damping characteristics of automotive components such as gaskets and grommets vary with the frequency of excitation as well as the level of preload. These structural properties also depend on the geometry and the level of confinement of the gasket. This capability allows the direct specification of such dynamic properties as quantified by the effective storage and loss moduli in the thickness-direction, as tabular functions of the frequency of excitation and the level of preload. The preload is quantified by the amount of closure in the base state about which the steady-state dynamic response is desired.

In determining the dynamic response of the gasket, the long-term elastic response is assumed to be defined by the nonlinear elastic model with damage. The steady-state dynamic response is assumed to be a perturbation about a base state defined by this elastic damage behavior at a certain value of closure. The viscoelastic response can be specified using two approaches, as discussed below.

Direct Specification of the Properties

The first approach involves direct (tabular) specification of the thickness-direction loss and storage moduli as functions of excitation frequency at different levels of closure.

Input File Usage: **VISCOELASTIC*, TYPE=TRACTION, PRELOAD=UNIAXIAL

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Elasticity->Viscoelastic**: **Domain: Frequency** and **Frequency: Tabular**

Specification of Properties in Terms of Ratios

The second approach allows the specification of the ratio of both the thickness-direction storage and the loss moduli to the long-term thickness-direction elastic modulus. These ratios can be specified as tabular functions of the excitation frequency but are assumed to be independent of the amount of closure. The actual storage or loss modulus at any given level of closure is computed by multiplying the appropriate ratio with the long-term elastic modulus at the current value of closure (of the base state). See *Frequency Domain Viscoelasticity* for a summary of the second approach in the context of continuum material viscoelastic properties (the approach used here is just a one-dimensional specialization of the more general approach presented there).

Input File Usage: **VISCOELASTIC*, TYPE=TRACTION

Abaqus/CAE Usage: Property module: material editor: **Mechanical->Elasticity->Viscoelastic**: **Domain: Frequency** and **Frequency: Tabular**

Defining the Contact Area for Average Contact Pressure Output

When the thickness-direction behavior of the gasket is defined in terms of force or force per unit length versus closure, Abaqus/Standard will provide the thickness-direction force or force per unit length as output variable S11. In this case you can define either a contact width or contact area versus closure curve that will be used to obtain the average "contact" pressure at each integration point as output variable CS11. This average pressure considers the changing contact area that occurs as a result of the deformation of a gasket, as shown in *Figure 1*. The closure used for input of this curve corresponds to the total mechanical closure, defined as the sum of the elastic, plastic, and creep closures.

When two- and three-dimensional link gasket elements are used, you should specify the contact area versus mechanical closure in tabular form. When axisymmetric link and three-dimensional line elements are used, you should specify the contact width versus mechanical closure in tabular form. A typical curve is shown in *Figure 8*.

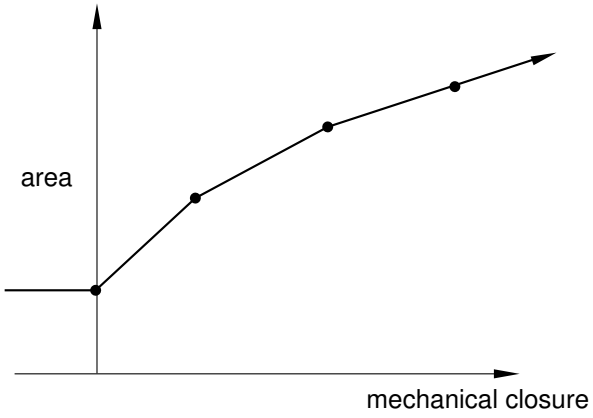


Figure 8: Specification of contact area versus mechanical closure for output of average pressure.

You must specify the area at zero closure, then the area at increasing closures. The area is constant when the mechanical closure is negative and is extrapolated from the slope computed from the last two user-specified data points if the closure reaches values that are greater than the last user-specified closure. Area versus closure curves can be provided as a function of temperature and field variables.

Input File Usage: **GASKET CONTACT AREA*, DEPENDENCIES

Abaqus/CAE Usage: Property module: material editor: **Other->Gasket->Gasket Thickness Behavior**: **Units: Force**, **Suboptions->Contact Area**

Specific Output for Directly Defined Gasket Behavior

Output variable E is usually used in Abaqus/Standard to output strain. For gasket elements with behavior defined by a gasket behavior model this output variable has thickness-direction and transverse shear components with units of displacement and membrane strains. Output variable NE is used to output an effective strain. The effective strain components are computed as follows:

$$\mathbf{NE11 = E11/(initial\ thickness - initial\ gap)\ for\ perturbation\ steps;\ otherwise}$$

$$\mathbf{NE11 = max(0, (E11 - initial\ gap)\ /(initial\ thickness - initial\ gap));\ and}$$

$$\mathbf{NE22 = E22;}$$

$$\mathbf{NE33 = E33;}$$

$$\mathbf{NE12 = E12/\ (initial\ thickness)\ ;}$$

$$\mathbf{NE13 = E13/(initial\ thickness);}$$

$$\mathbf{NE23 = E23.}$$

The output variables THE, PE, or CE can also be used for gasket elements to output generalized thermal strains, plastic strains, or creep strains, respectively.

For all stress/strain output variables the 11-component refers to the through-thickness direction; the 22-, 33- and 23-components refer to two direct and one shear membrane component, respectively; the remaining 12- and 13-components refer to the transverse shear components. For details about these definitions, see *About Gasket Elements*.

The output of the elastic strain energy (output variable ALLSE) also contains the energy due to damage or change in elasticity as a function of plasticity. Therefore, this energy is usually not fully recoverable.

References

- Zubeck, M. W., and R. S. Marlow, "Local-Global Finite Element Analysis for Cam Cover Noise Reduction," *Society of Automotive Engineering, Inc.*, no. SAE 2003–01–1725, 2003.

Defining a Thermal Interaction for a Gasket Element

Products: Abaqus/Standard Abaqus/CAE

References:

- *Three-Dimensional Gasket Element Library*
- *GAPCON*
- **GAP CONDUCTANCE*

Overview

You can define thermal interaction between the top and bottom surfaces of a coupled temperature-displacement gasket element.

A thermal interaction between the top and bottom surfaces of a three-dimensional coupled temperature-displacement gasket element:

- can be included in a thermal-stress analysis (*Fully Coupled Thermal-Stress Analysis*);
- can involve conductive heat transfer between the top and bottom surfaces; and
- can be specified for gasket behavior defined by a material model or a gasket behavior model.

Specifying Thermal Conductance of a Gasket Element

The thermal conductance between the top and bottom surfaces of a three-dimensional coupled temperature-displacement gasket element can be specified for gasket behavior defined by a material model or a gasket behavior model.

Input File Usage: Use the following options to define the thermal conductance with a material model:

**GASKET SECTION*, ELSET=*name*, MATERIAL=*name*
**MATERIAL*, NAME=*name*
**GAP CONDUCTANCE*

Use the following options to define the thermal conductance with a gasket behavior model:

**GASKET SECTION*, ELSET=*name*, BEHAVIOR=*name*
**GASKET BEHAVIOR*, NAME=*name*
**GAP CONDUCTANCE*

Abaqus/CAE Usage: Use the following option to define the thermal conductance with a material model:

Property module: material editor: **Thermal->Gap Conductance**

Modeling Conductance between the Top and Bottom Surfaces of a Gasket Element

The conductive heat transfer between the top and bottom surfaces of a gasket element is assumed to be defined by

$$q = k(\theta_A - \theta_B),$$

where q is the heat flux per unit area crossing the gasket from point A on the top surface to point B on the bottom surface, θ_A and θ_B are the temperatures of the points on the surfaces, and k is the gap conductance. Point A is a node on the top surface; and point B is the corresponding node on the bottom surface.

You can define k directly or, in Abaqus/Standard, in user subroutine *GAPCON*.

Defining Gap Conductance Directly

When defining k directly, define it as

$$k = k\left(c, p, \overline{\theta}, \overline{|\dot{m}|}, \overline{f}_\gamma\right),$$

where

c

is the closure between A and B,

p

is the pressure transmitted across the gasket between A and B,

$\overline{\theta} = \frac{1}{2}\left(\theta_A + \theta_B\right)$

is the average of the surface temperatures at A and B,

$\overline{|\dot{m}|}$

is the average of the magnitudes of the mass flow rates per unit area. This variable is not considered here and should be set as zero.

$\overline{f}_\gamma = \frac{1}{2}\left(f_\gamma^A + f_\gamma^B\right)$

is the average of any predefined field variables at A and B.

Defining Gap Conductance as a Function of Closure

You can create a table of data defining the dependence of k on the variables listed above. The default in Abaqus is to make k a function of the closure c. When k is a function of closure, c, the tabular data must start at zero closure and define k as c increases. At least two pairs of $k - c$ points must be given to define k as a function of the closure. The value of k drops to zero immediately after the last data point, so there is no heat conductance when the closure is greater than the value corresponding to the last data point.

Input File Usage: **GAP CONDUCTANCE*
k, c, $\overline{\theta}$

Abaqus/CAE Usage: Property module: material editor: **Thermal->Gap Conductance**: enter table data

Defining Gap Conductance as a Function of Contact Pressure

You can define k as a function of the pressure, p. When k is a function of pressure in gasket elements, the tabular data must start at zero pressure and define k as p increases. The value of k remains constant for pressures outside of the interval defined by the data points.

Input File Usage: **GAP CONDUCTANCE*, PRESSURE
k, p, $\overline{\theta}$

Abaqus/CAE Usage: Property module: material editor: **Thermal->Gap Conductance**: **Use pressure-dependent data**

Gap Conductance as a Function of Both Closure and Contact Pressure

If both closure-dependent and pressure-dependent conductances are specified, the pressure-dependent curves are used to evaluate the conductance.

Defining Gap Conductance to Be a Function of Predefined Field Variables

In addition to the dependencies mentioned previously, the gap conductance can be dependent on any number of predefined field variables, $\overline{f}_\gamma$. To make the gap conductance depend on field variables, at least two data points are required for each field variable value.

Input File Usage: **GAP CONDUCTANCE*, DEPENDENCIES=n
k, c, $\overline{\theta}$, $\overline{|\dot{m}|}$, $\overline{f}_\gamma$

Abaqus/CAE Usage: Property module: material editor: **Thermal->Gap Conductance**: **Number of field variables**: n

Defining the Gap Conductance Using User Subroutine GAPCON

In Abaqus/Standard$k$ can be defined in user subroutine *GAPCON*. In this case there is greater flexibility in specifying the dependencies of k. It is no longer necessary to define k as a function of the average of the two surface's temperatures, mass flow rates, or field variables:

$$k = k\left(c, p, \theta_A, \theta_B, |\dot{m}|_A, |\dot{m}|_B, f_\gamma^A, f_\gamma^B\right).$$

The mass flow rates in user subroutine *GAPCON* are not used to model conductance in gasket elements, and the variables should be set to zero.

Input File Usage: **GAP CONDUCTANCE*, USER

Abaqus/CAE Usage: Property module: material editor: **Thermal->Gap Conductance**: User-defined

Two-Dimensional Gasket Element Library

Products: Abaqus/Standard Abaqus/CAE

Overview

References:

- *About Gasket Elements*
- *Choosing a Gasket Element*
- **GASKET SECTION*

Element Types

Link Elements

GK2D2	2-node, two-dimensional gasket element
GK2D2N	2-node, two-dimensional gasket element with thickness-direction behavior only

Active Degrees of Freedom

1 for gasket elements with thickness-direction behavior only.

1, 2 for other gasket elements.

Additional Solution Variables

None.

General Elements

GKPS4	4-node, plane stress gasket element
GKPE4	4-node, plane strain gasket element
GKPS4N	4-node, two-dimensional gasket element with thickness-direction behavior only
GKPS6	6-node, plane stress gasket element
GKPE6	6-node, plane strain gasket element
GKPS6N	6-node, two-dimensional gasket element with thickness-direction behavior only

Active Degrees of Freedom

1 for gasket elements with thickness-direction behavior only.

1, 2 for other gasket elements.

Additional Solution Variables

None.

Nodal Coordinates Required

X, Y

Element Property Definition

You must define the element's cross-sectional area (for link elements) or out-of-plane width (for general elements), initial gap, and initial void.

You can specify the thickness direction as part of the gasket section definition or by specifying a normal direction at the nodes; you can specify the element thickness as part of the gasket section definition. Otherwise, Abaqus/Standard will calculate the thickness direction. For link elements the thickness direction is the direction from the first to the second node and the thickness is the distance between the nodes. For general elements the thickness direction is based on the midsurface of the element and the thicknesses at the integration points are based on the nodal positions. See *Defining the Gasket Element's Initial Geometry* for more details.

Input File Usage: **GASKET SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Gasket** as the section **Type**

Element-Based Loading

None.

Element Output

GK2D2 Elements

S11 Pressure or thickness-direction force in the gasket element.

CS11 Contact pressure in the gasket element (only available if S11 is the force in the gasket element and the gasket response is not defined using a material model).

S12 Shear stress or shear force.

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

E12 Shear motion if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain in the gasket element.

NE12 Effective shear strain in the gasket element.

GK2D2N Elements

S11 Pressure or thickness-direction force in the gasket element.

CS11 Contact pressure in the gasket element (only available if S11 is the force in the gasket element and the gasket response is not defined using a material model).

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain in the gasket element.

General Elements with Thickness-Direction Behavior Only

S11 Pressure in the gasket element.

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain in the gasket element.

Other General Elements

S11 Pressure in the gasket element.

S22 Direct membrane stress.

S33 Direct membrane stress (only available for plane strain elements).

S12 Shear stress.

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

E22 Direct membrane strain.

E33 Direct membrane strain (only available for plane strain elements).

E12 Shear motion if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain in the gasket element.

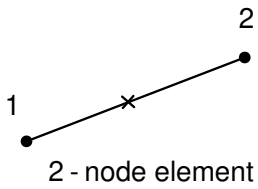
NE22 Direct membrane strain.

NE33 Direct membrane strain (only available for plane strain elements).

NE12 Effective shear strain.

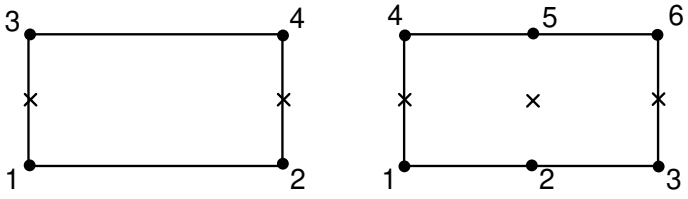
Node Ordering and Integration Point Numbering

Link Elements



2 - node element

General Elements



4 - node element

6 - node element

Three-Dimensional Gasket Element Library

Products: Abaqus/Standard Abaqus/CAE

Overview

This section provides a reference to the three-dimensional gasket elements available in Abaqus/Standard.

References:

- *About Gasket Elements*
- *Choosing a Gasket Element*
- **GASKET SECTION*

Element Types

Link Elements

GK3D2	2-node, three-dimensional gasket element
GK3D2N	2-node, three-dimensional gasket element with thickness-direction behavior only

Active Degrees of Freedom

1 for gasket elements with thickness-direction behavior only.

1, 2, 3 for other gasket elements.

Additional Solution Variables

None.

Line Elements

GK3D4L	4-node, three-dimensional line gasket element
GK3D4LN	4-node, three-dimensional line gasket element with thickness-direction behavior only
GK3D6L	6-node, three-dimensional line gasket element
GK3D6LN	6-node, three-dimensional line gasket element with thickness-direction behavior only

Active Degrees of Freedom

1 for gasket elements with thickness-direction behavior only.

1, 2, 3 for other gasket elements.

Additional Solution Variables

None.

Area Elements

GK3D6	6-node, three-dimensional gasket element
GK3D6N	6-node, three-dimensional gasket element with thickness-direction behavior only
GK3D8	8-node, three-dimensional gasket element
GK3D8N	8-node, three-dimensional gasket element with thickness-direction behavior only
GK3D12M	12-node, three-dimensional gasket element
GK3D12MN	12-node, three-dimensional gasket element with thickness-direction behavior only
GK3D18	18-node, three-dimensional gasket element
GK3D18N	18-node, three-dimensional gasket element with thickness-direction behavior only

Active Degrees of Freedom

1 for gasket elements with thickness-direction behavior only.

1, 2, 3 for other gasket elements.

Additional Solution Variables

None.

Coupled Temperature-Displacement Area Elements

GK3D6T	6-node, three-dimensional displacement and temperature gasket element
GK3D6TN	6-node, three-dimensional displacement and temperature gasket element with thickness-direction behavior only
GK3D8T	8-node, three-dimensional displacement and temperature gasket element
GK3D8TN	8-node, three-dimensional displacement and temperature gasket element with thickness-direction behavior only

Active Degrees of Freedom

1, 11 for gasket elements with thickness-direction behavior only.

1, 2, 3, 11 for other gasket elements.

Additional Solution Variables

None.

Nodal Coordinates Required

X, Y, Z

Element Property Definition

You must define the element's initial gap and initial void, as well as the cross-sectional area (for link elements) or width (for line elements).

You can specify the thickness direction as part of the gasket section definition or by specifying a normal direction at the nodes; you can specify the element thickness as part of the gasket section definition. Otherwise, Abaqus/Standard will calculate the thickness direction and the thickness. For link elements the thickness direction is the direction from the first to the second node and the thickness is the distance between the nodes. For line elements the thickness direction is the direction from the bottom node to the top node associated with the integration point and the thicknesses are the distances between these same bottom and top nodes. For area elements the thickness direction is based on the midsurface of the element and the thicknesses at the integration points are based on the nodal positions. See *Defining the Gasket Element's Initial Geometry* for more details.

Input File Usage: **GASKET SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Gasket** as the section **Type**

Element-Based Loading

None.

Element Output

GK3D2 Elements

S11 Pressure or thickness-direction force in the gasket element.

CS11 Contact pressure in the gasket element (only available if S11 is a force and the gasket response is not defined using a material model).

S12 Shear stress or shear force.

S13 Shear stress or shear force.

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

E12 Shear motion if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

E13 Shear motion if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain in the gasket element.

NE12 Effective shear strain.

NE13 Effective shear strain.

GK3D2N Elements

S11 Pressure or thickness-direction force in the gasket element.

CS11 Contact pressure in the gasket element (only available if S11 is a force and the gasket response is not defined using a material model.)

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain in the gasket element.

Line Elements with Thickness-Direction Behavior Only

S11 Pressure or thickness-direction force per unit length in the gasket element.

CS11 Contact pressure in the gasket element (only available if S11 is a force per unit length and the gasket response is not defined using a material model).

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain in the gasket element.

Other Line Elements

S11 Pressure or thickness-direction force per unit length in the gasket element.

CS11 Contact pressure in the gasket element (only available if S11 is a force per unit length and the gasket response is not defined using a material model).

S22 Direct membrane stress.

S12 Shear stress or shear force per unit length.

S13 Shear stress or shear force per unit length.

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

E22 Direct membrane strain.

E12 Shear motion if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

E13 Shear motion if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain in the gasket element.

NE22 Direct membrane strain.

NE12 Effective shear strain.

NE13 Effective shear strain.

Area Elements with Thickness-Direction Behavior Only

S11 Pressure in the gasket element.

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain in the gasket element.

Other Area Elements

S11 Pressure in the gasket element.

S22 Direct membrane stress.

S33 Direct membrane stress.

S12 Transverse shear stress.

S13 Transverse shear stress.

S23 Membrane shear stress.

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

E22 Direct membrane strain.

E33 Direct membrane strain.

E12 Transverse shear motion if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

E13 Transverse shear motion if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

E23 Membrane shear strain.

NE11 Effective thickness-direction strain in the gasket element.

NE22 Direct membrane strain.

NE33 Direct membrane strain.

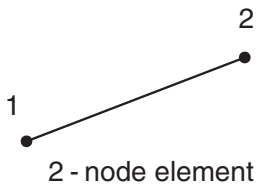
NE12 Effective shear strain.

NE13 Effective shear strain.

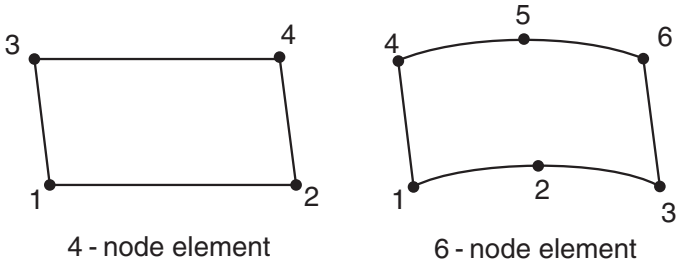
NE12 Membrane shear strain.

Node Ordering and Face Numbering on Elements

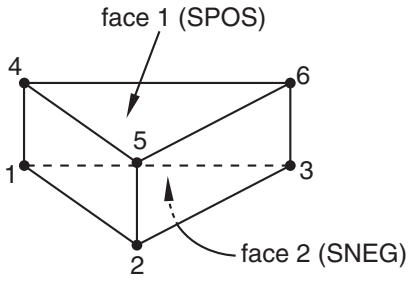
Link Elements

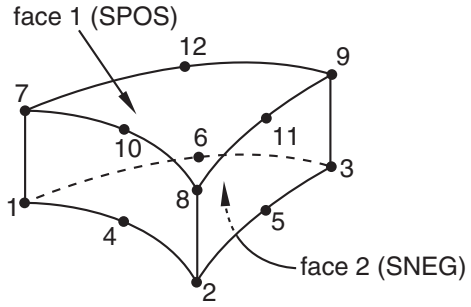


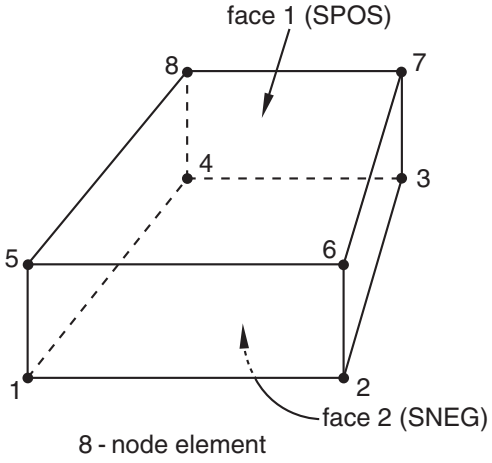
Line Elements



Area Elements







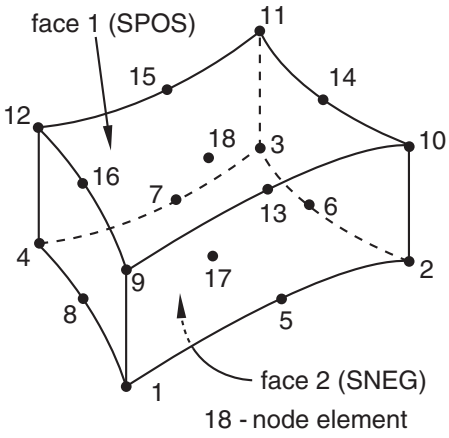


Table 1: 6-node element

Face 1 (SPOS)	4 – 6 – 5 face
Face 2 (SNEG)	1 – 2 – 3 face

Table 2: 8-node element

Face 1 (SPOS)	5 – 8 – 7 – 6 face

Face 2 (SNEG)	1 – 2 – 3 – 4 face

Table 3: 12-node element

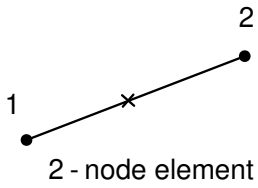
Face 1 (SPOS)	7 – 12 – 9 – 11 – 8 – 10 face
Face 2 (SNEG)	1 – 4 – 2 – 5 – 3 – 6 face

Table 4: 18-node element

Face 1 (SPOS)	9 – 16 – 12 – 15 – 11 – 14 – 10 – 13 face
Face 2 (SNEG)	1 – 5 – 2 – 6 – 3 – 7 – 4 – 8 face

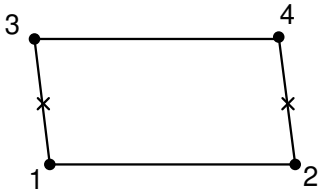
Numbering of Integration Points for Output

Link Elements

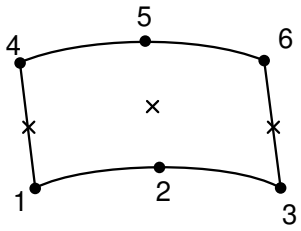


2 - node element

Line Elements

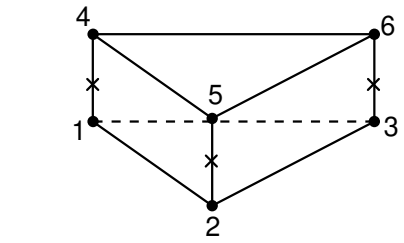


4 - node element

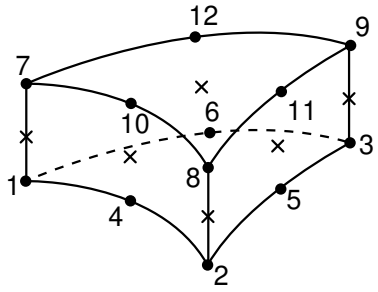


6 - node element

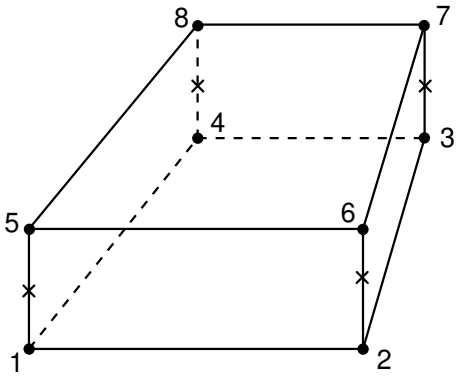
Area Elements



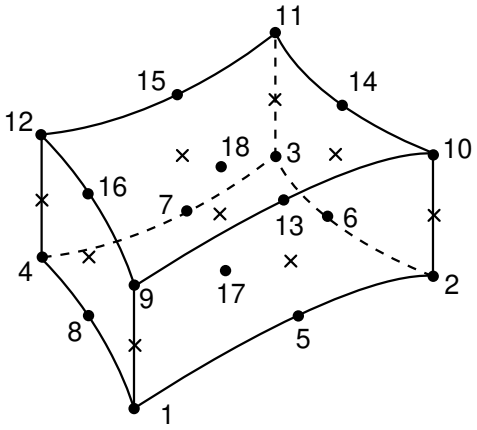
6 - node element



12 - node element



8 - node element



18 - node element

Integration points are indicated with an X and have the same numbers as the bottom face nodes, except that the point between nodes 17 and 18 in the 18-node gasket element is integration point number 9.

Axisymmetric Gasket Element Library

Products: Abaqus/Standard Abaqus/CAE

Overview

References:

- *About Gasket Elements*
- *Choosing a Gasket Element*
- **GASKET SECTION*

Element Types

Link Elements

GKAX2	2-node, axisymmetric gasket element
GKAX2N	2-node, axisymmetric gasket element with thickness-direction behavior only

Active Degrees of Freedom

1 for gasket elements with thickness-direction behavior only.

1, 2 for other gasket elements.

Additional Solution Variables

None.

General Elements

GKAX4	4-node, axisymmetric gasket element
GKAX4N	4-node, axisymmetric gasket element with thickness-direction behavior only
GKAX6	6-node, axisymmetric gasket element
GKAX6N	6-node, axisymmetric gasket element with thickness-direction behavior only

Active Degrees of Freedom

1 for gasket elements with thickness-direction behavior only.

1, 2 for other gasket elements.

Additional Solution Variables

None.

Nodal Coordinates Required

X, Y

Element Property Definition

You must define the element's initial gap and initial void. In addition, for link elements you must define the element's width.

You can specify the thickness direction as part of the gasket section definition or by specifying a normal direction at the nodes; you can specify the element thickness as part of the gasket section definition. Otherwise, Abaqus/Standard will calculate the thickness direction and the thickness. For link elements the thickness direction is the direction from the first to the second node and the thickness is the distance between the nodes. For general elements the thickness direction is based on the midsurface of the element and the thickness at the integration points are based on the nodal positions. See *Defining the Gasket Element's Initial Geometry* for more details.

Input File Usage:

**GASKET SECTION*

Abaqus/CAE Usage: Property module: **Create Section**: select **Other** as the section **Category** and **Gasket** as the section **Type**

Element-Based Loading

None.

Element Output

GKAX2 Elements

S11 Pressure or thickness-direction force per unit length in the gasket element.

CS11 Contact pressure in the gasket element (only available if S11 is a force per unit length and the gasket response is not defined using a material model).

S22 Hoop stress.

S12 Shear stress or shear force per unit length.

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

E22 Hoop strain.

E12 Shear motion if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain.

NE22 Hoop strain.

NE12 Effective shear strain.

GKAX2N Elements

S11 Pressure or thickness-direction force per unit length in the gasket element.

CS11 Contact pressure in the gasket element (only available if S11 is a force per unit length and the gasket response is not defined using a material model).

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain.

General Elements with Thickness-Direction Behavior Only

S11 Pressure in the gasket element.

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain.

Other General Elements

S11 Pressure in the gasket element.

S22 Direct membrane stress.

S33 Hoop stress.

S12 Shear stress.

E11 Gasket closure if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

E22 Direct membrane strain.

E33 Hoop strain.

E12 Shear motion if the gasket response is defined directly using a gasket behavior model; strain if the gasket response is defined using a material model.

NE11 Effective thickness-direction strain.

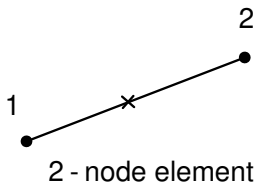
NE22 Direct membrane strain.

NE33 Direct membrane strain.

NE12 Effective shear strain.

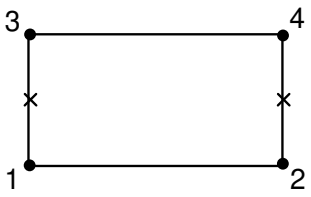
Node Ordering and Integration Point Numbering

Link Elements

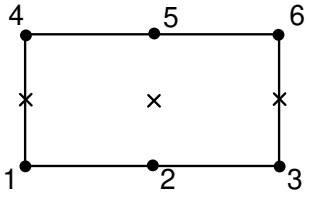


2 - node element

General Elements



4 - node element



6 - node element

Line Spring Elements

In this section:

- *Line Spring Elements for Modeling Part-through Cracks in Shells*
- *Line Spring Element Library*

Line Spring Elements for Modeling Part-through Cracks in Shells

Products: Abaqus/Standard

References:

- *Line Spring Element Library*
- **SHELL SECTION*
- **SURFACE FLAW*

Overview

Line spring elements:

- are used to evaluate part-through cracks (flaws) in shells inexpensively;
- are used together with shell elements;
- can be used with elastic or elastic-plastic (isotropic hardening, Mises yield) material behavior;
- do not include thermal strain effects;
- are written for small-displacement analysis only (large-rotation effects are not included);
- are not available in linear perturbation steps;
- use quite significant approximations (especially in the elastic-plastic case) and should, therefore, be used with care;
- do not provide useful results for crack depths less than 2% or greater than 95% of the shell thickness; and
- will not yield accurate results at the ends of the flaws or locations where the flaw depth varies rapidly with position, due to the three-dimensional nature of the solution in such areas.

Typical Applications

Line spring elements provide inexpensive evaluation of part-through cracks in shells. The basic concept is that these elements introduce the local solution, dominated by the singularity at the crack tip, into a shell model of the uncracked geometry. This is accomplished by allowing an additional freedom in the model along the line of the crack, this freedom being provided by the line spring elements, as indicated in *Figure 1*.

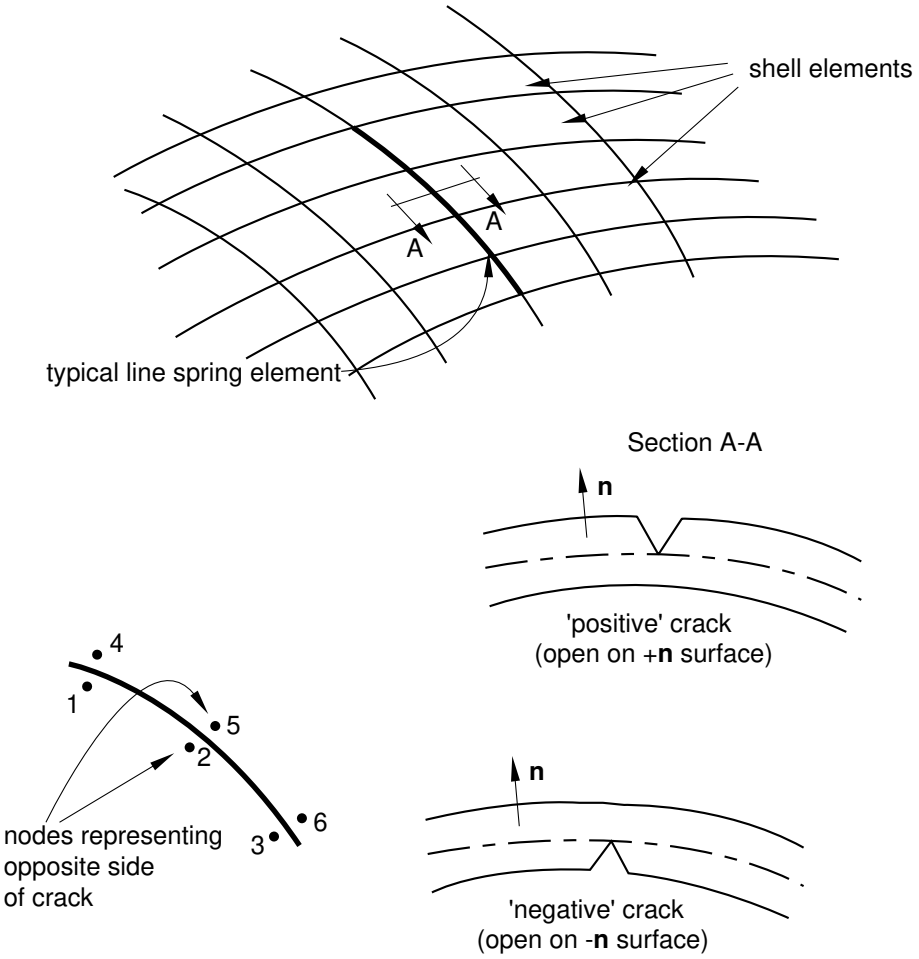


Figure 1: Line spring models.

The compliance of the line spring with respect to these additional freedoms embeds the local solution in the global response. From the relative displacements and rotations conjugate to that compliance, Abaqus/Standard computes and prints out the *J*-integral and, in the linear case, stress intensity factors at integration points in the line spring elements. Because the elements are simple, the analysis is not significantly more expensive than a shell analysis of the uncracked geometry. The results provide acceptable accuracy for many common applications.

See *Line spring elements* for details of the theory behind these elements.

Choosing an Appropriate Element

Two versions of the element are provided—both are intended for use with the second-order shell elements (S8R, S8R5, S9R5). Line spring element LS6 is for general cases, while line spring element LS3S is for use when the flaw lies on a symmetry plane and only one side of the symmetry plane is modeled.

Defining the Elements Section Properties

You must associate the shell section properties with a set of line spring elements.

Input File Usage: **SHELL SECTION*, ELSET=*name*

Defining a Constant Section Thickness

You can define a constant section thickness for the line spring element as part of the shell section definition.

Input File Usage: **SHELL SECTION*
shell thickness

Defining a Variable Section Thickness

Alternatively, you can define a line spring element with continuously varying thickness and specify the thickness of the line spring element at the nodes. In this case any constant section thickness you specify will be ignored, and the line spring thickness will be interpolated from the nodes (see *Nodal Thicknesses*). The thickness must be defined at all nodes connected to the element.

Input File Usage: Use both of the following options:

**SHELL SECTION*, NODAL THICKNESS
**NODAL THICKNESS*

Assigning a Material Definition to a Set of Line Spring Elements

You must associate a material definition with each shell section definition.

Line spring elements can be used with isotropic elastic or elastic-plastic (isotropic hardening, Mises yield) material behavior (*Linear Elastic Behavior* and *Classical Metal Plasticity*); these are the only material behavior definitions that are relevant to these elements. The elastic behavior must be isotropic. Plasticity is included for Mode I (crack opening) response only; an elastic-plastic analysis will be accurate only when Mode I behavior dominates.

The same material must be used through the section: a layered section cannot be defined with a line spring. Thermal strain effects are not included in the line spring elements; however, most of the thermal strain occurs in the shell, so this is not important in many cases (it is within the approximation made by line springs).

Input File Usage: **SHELL SECTION*, ELSET=*name*, MATERIAL=*name*

Defining the Flaw

The flaw is defined by specifying its depth at each node along the crack front. You must identify whether the crack originates from the positive or negative surface of the shell (the positive surface is located a positive distance along the surface normal from the shell's middle surface, as shown in *Figure 1*).

At a point where the surface flaw depth is very small or zero, the compliance of the line spring element is also very small. To avoid numerical problems when a small compliance is inverted to form a stiffness, the minimum surface flaw depth used by Abaqus/Standard is 2% of the thickness specified for the line spring element, even if you specify a smaller surface flaw depth. If you want to constrain the two nodes where the surface flaw depth is zero to have the same displacements, you should tie the nodes together with a linear constraint equation or a multi-point constraint (*About Kinematic Constraints*). This is normally not required.

Input File Usage: **SURFACE FLAW*, SIDE=POSITIVE or NEGATIVE
node number or node set label, *crack depth*
...

Defining the Shell Model That Contains the Flaw

You must specify the uncracked thickness of the shell in the section definition. The geometry of the shell at the flaw (coordinates and surface normals) is given in the usual way.

Including the Effects of Pressure Loading on the Crack Faces

Cracks often occur on surfaces that are subjected to pressure; to include the effect of such loading on the crack faces, suitable distributed loading types are provided. These loading types are not intended for elastic-plastic line springs because the nodal equivalent forces calculated for the pressures are based on superposition methods that are valid only in the linear elastic case.

J-Integral Output

If the material is linear elastic only, the *J*-integral value and the stress intensity factors are output; for the elastic-plastic case local values of J^{el} and J^{pl} are provided as well as their sum into a single *J* value. In this case the *J* values will have acceptable accuracy only if J^{pl} is much larger than J^{el}. See *Line spring elements* for further details.

Line Spring Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the line spring elements available in Abaqus/Standard.

References:

- *Line Spring Elements for Modeling Part-through Cracks in Shells*
- **SHELL SECTION*
- **SURFACE FLAW*

Element Types

LS6	6-node general second-order line spring
LS3S	3-node second-order line spring for use on a symmetry plane

Active Degrees of Freedom

1, 2, 3, 4, 5, 6

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*, *Z* required at each node and, optionally, N_x, N_y, N_z (direction cosines of the normal to the shell) at each node.

A user-defined normal definition (see *Normal Definitions at Nodes*) can also be used to specify N_x, N_y, N_z. If these are not specified, they are constructed as for all other shell elements—by averaging over the shell elements attached to each node.

Element Property Definition

The only element property used is the thickness; the number of integration points is ignored, since the elements work on the basis of section properties.

Input File Usage: Use the following option to define line spring element properties:

**SHELL SECTION*

Use the following option to define the depth of the crack as a function of position:

**SURFACE FLAW*

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*.

Three Gauss points are used for crack face pressure loading.

Load ID (*DLOAD): HP
Units: FL^{-2}
Description: Hydrostatic surface pressure on the crack faces, with magnitude varying linearly with the global *Z*-direction.

Load ID (*DLOAD): P
Units: FL^{-2}
Description: Surface pressure on the crack faces.

Element Output

Nodes 1, 2, and 3 on the element define side *B* and nodes 4, 5, and 6 define side *A* (see *Figure 1*). The sign of the crack is defined by the surface of the shell from which the crack originates, which you identify when you define the depth of the crack (see *Line Spring Elements for Modeling Part-through Cracks in Shells*). If the crack originates from the positive surface of the shell, *sign*(crack)=1.0; if the crack originates from the negative surface of the shell, *sign*(crack)=−1.0.

The vector $\mathbf{q}$ is defined by the right-hand rule from the cross product of the tangent, $\mathbf{t}$, which is positive going from node 1 to node 3 of the element, and the normal, $\mathbf{n}$, defined when the coordinates are given (or by a user-defined normal definition). For element type LS3S the vector $\mathbf{q}$ must point into the model (away from the symmetry plane). For element type LS6 the vector $\mathbf{q}$ must point from side *A* to side *B*.

"Strains"

E11
Mode I opening displacement, $(u_B - u_A) \cdot \underline{\mathbf{q}}$

E22
Mode I opening rotation, $(\phi_B - \phi_A) \cdot \mathbf{t} \times sign\,(\mathbf{crack})$

The following strains exist only for LS6:

E33
Mode II through thickness shear, $(\mathbf{u}_B - \mathbf{u}_A) \cdot \mathbf{n}$

E12
Mode II rotation, $(\phi_B - \phi_A) \cdot \mathbf{n}$ (this strain plays no role)

E13
Mode III antiplane shear, $(\mathbf{u}_B - \mathbf{u}_A) \cdot \mathbf{t} \times sign\,(\mathbf{crack})$

E23

Mode III opening rotation, $(\boldsymbol{\phi}_B - \boldsymbol{\phi}_A) \cdot \mathbf{q}$

The conjugate forces and moments are available by requesting “stress” output.

The *J*-integral is provided at each integration point. If elastic-plastic material behavior is defined, the elastic and plastic parts of *J* are provided. The stress intensity factors, *K*, are also provided corresponding to the elastic parts of *J*.

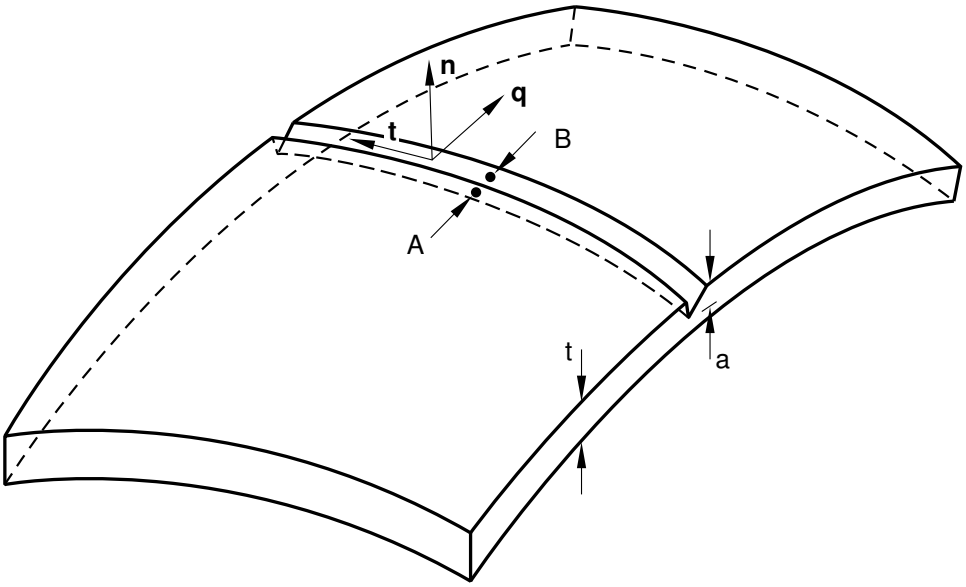
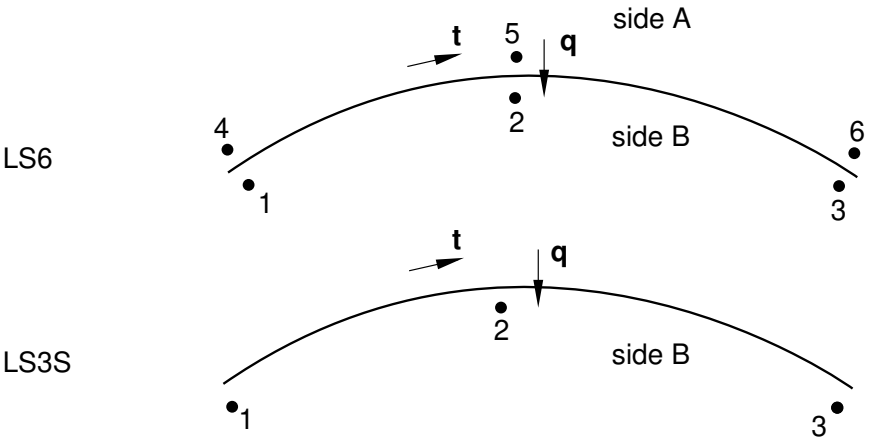


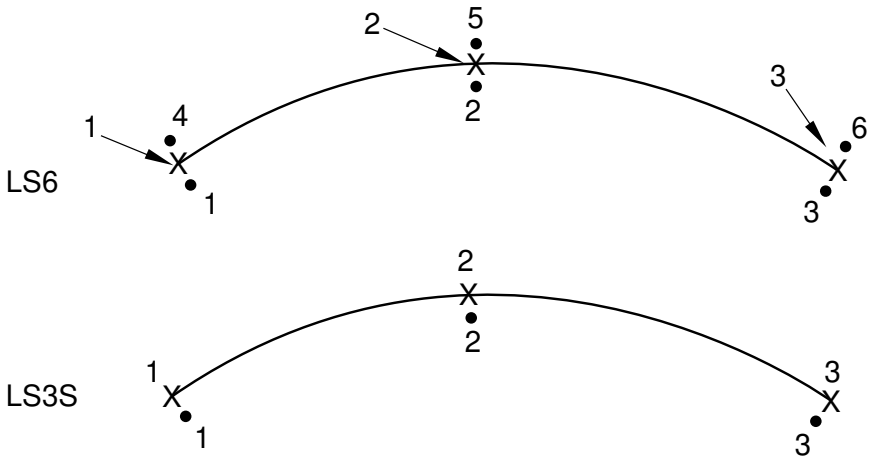
Figure 1: Notation for line spring strains.

Nodes Associated with the Element



Numbering of Integration Points for Output

Three points (these points are at the nodes) are used for integration and element output.


LS6
1
4
1
2
5
2
3
6
3
LS3S
1
1
2
2
3
3

Pipe-Soil Elements

In this section:

- *Pipe-Soil Interaction Elements*
- *Pipe-Soil Interaction Element Library*

Pipe-Soil Interaction Elements

Products: Abaqus/Standard

References:

- *Pipe-Soil Interaction Element Library*
- **PIPE-SOIL INTERACTION*
- **PIPE-SOIL STIFFNESS*

Overview

The pipe-soil interaction elements in Abaqus/Standard:

- can be used to model the interaction between a buried pipeline and the surrounding soil;
- must be used with beam elements, pipe, or elbow elements (see *About Beam Modeling* and *Pipes and Pipebends with Deforming Cross-Sections: Elbow Elements*); and
- can have linear or nonlinear constitutive behavior.

Pipe Foundation Elements

Abaqus/Standard provides two-dimensional (PSI24 and PSI26) and three-dimensional (PSI34 and PSI36) pipe-soil interaction elements for modeling the interaction between a buried pipeline and the surrounding soil.

The pipeline itself is modeled with any of the beam, pipe, or elbow elements in the Abaqus/Standard element library (see *About Beam Modeling* and *Pipes and Pipebends with Deforming Cross-Sections: Elbow Elements*). The ground behavior and soil-pipe interaction are modeled with the pipe-soil interaction (PSI) elements. These elements have only displacement degrees of freedom at their nodes. One side or edge of the element shares nodes with the underlying beam, pipe, or elbow element that models the pipeline (see *Figure 1*). The nodes on the other edge represent a far-field surface, such as the ground surface, and are used to prescribe the far-field ground motion via boundary conditions together with amplitude references as needed.

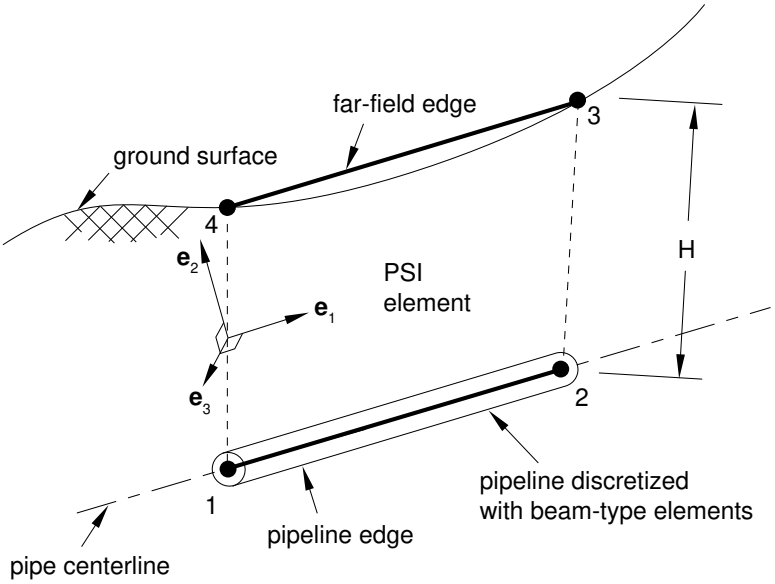


Figure 1: Pipe-soil interaction model.

The far-field side and the side that shares nodes with the pipeline are defined by the element connectivity. Care must be taken in attaching the underlying elements to the correct edge of the PSI element, since the connectivity of the pipe-soil element determines the local coordinate system as defined below, and the depth, H, of the pipeline below the ground surface. The depth below the surface is measured along the edge of the PSI element as shown in *Figure 1* and is updated during geometrically nonlinear analysis.

It is important to note that PSI elements do not discretize the actual domain of the surrounding soil. The extent of the soil domain is reflected through the stiffness of the elements, which is defined by the constitutive model as described later.

The pipe-soil interaction model does not include the density of the surrounding soil medium. Mass can be associated with the model by applying concentrated MASS elements (see *Point Masses*) at the nodes of the pipe-soil interaction elements if needed.

Assigning the Pipe-Soil Interaction Behavior to a PSI Element

You must assign the pipe-soil interaction behavior to a set of pipe-soil interaction elements.

Input File Usage: Use the following option to assign the pipe-soil interaction behavior to a particular element set:

**PIPE-SOIL INTERACTION*, ELSET=*name*

Use the following option immediately after the **PIPE-SOIL INTERACTION* option to define the stiffness behavior for the element set:

**PIPE-SOIL STIFFNESS*

Kinematics and Local Coordinate System

The deformation of the pipe-soil interaction element is characterized by the relative displacements between the two edges of the element. When the element is "strained" by the relative displacements, forces are applied to the pipeline nodes. These forces can be a linear (elastic) or nonlinear (elastic-plastic) function of the "strains," depending on the type of constitutive model used for the element. Positive "strains" are defined by

$$\varepsilon_{ii} = \Delta u \cdot e_i \qquad \text{(no summation)},$$

where

$$\Delta \mathbf{u} = \mathbf{u}^f - \mathbf{u}^p$$

are the relative displacements between the two edges ($\mathbf{u}^f$ are the far-field displacements, and $\mathbf{u}^p$ are the pipeline displacements), $\mathbf{e}_i$ are local directions, and the index i (=1, 2, 3) refers to the three local directions. For two-dimensional elements only the in-plane components of strain ε_{11}, ε_{22} exist. For three-dimensional elements all three strain components ε_{11}, ε_{22}, and ε_{33} exist.

The local orientation system is defined by three orthonormal directions: $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_3$. The default local directions are defined so that $\mathbf{e}_1$ is the direction along the pipeline (axial direction), $\mathbf{e}_3$ is the direction normal to the plane of the element (transverse horizontal direction), and $\mathbf{e}_2 = \mathbf{e}_3 \times \mathbf{e}_1$ is the direction in the plane of the element that defines the transverse vertical behavior. Positive default directions are defined so that $\mathbf{e}_1$ points toward the second pipeline node and $\mathbf{e}_2$ points from the pipeline edge toward the far-field edge, as shown in *Figure 1*. You can also define these local directions by specifying a local orientation (*Orientations*) for the pipe-soil interaction.

In a large-displacement analysis the local coordinate system rotates with the rigid body motion of the underlying pipeline. In a small-displacement analysis the local system is defined by the initial geometry of the PSI element and remains fixed in space during the analysis.

Input File Usage: Use the following option to associate a local orientation with a pipe-soil interaction behavior:

**PIPE-SOIL INTERACTION*, ORIENTATION=*name*

Constitutive Models

The constitutive behavior for a pipe-soil interaction is defined by the force per unit length, or "stress," at each point along the pipeline, q_i, caused by relative displacement or "strain," ε_{jj}, between that point and the point on the far-field surface:

$$q_i = q_i\left(\varepsilon_{jj}, s_\alpha, f_\beta\right),$$

where s_α are state variables (such as plastic strains), and f_β are temperatures and/or field variables.

You can define these q_i relationships quite generally by programming them in user subroutine *UMAT*. Alternatively, you can define the relationships by specifying the data directly. In this case the assumption is that the foundation behavior is separable:

$$q_i = q_i\left(\varepsilon_{ii}, s_\alpha, f_\beta\right),$$

in which case each of the independent relationships must be defined separately. Abaqus/Standard assumes, by default, that these relationships are symmetric about the origin (as is generally appropriate for the axial and transverse horizontal motions). However, you may give a nonsymmetric behavior for any of the three relative motions (this is usually the case in the vertical direction when the pipeline is not buried too deeply). These models assume that positive "strains" give rise to forces on the pipe that act along the positive directions of the local coordinate system.

Specifying the Constitutive Behavior with a User Subroutine

To define the q_i relationships quite generally, you can program them in user subroutine *UMAT*.

Input File Usage: **PIPE-SOIL STIFFNESS*, TYPE=USER

Specifying the Constitutive Behavior Directly

Two methods are provided for specifying constitutive behavior data directly. One method is to define the q_i relationships directly in tabular (piecewise linear) form. The other method is to use ASCE formulae. Forms of these relationships suitable for use with sands and clays are defined in the ASCE Guidelines for the Seismic Design of Oil and Gas Pipeline Systems.

Specifying the Constitutive Behavior Directly Using Tabular Input

You can define a linear or nonlinear constitutive model with different behavior in tension and compression using tabular input.

Linear Model

To define a linear constitutive model, you specify the stiffness as a function of temperature and field variables (see *Figure 2*). You can enter different values for positive and negative "strain." Abaqus/Standard assumes, by default, that the relationship is symmetric about the origin.

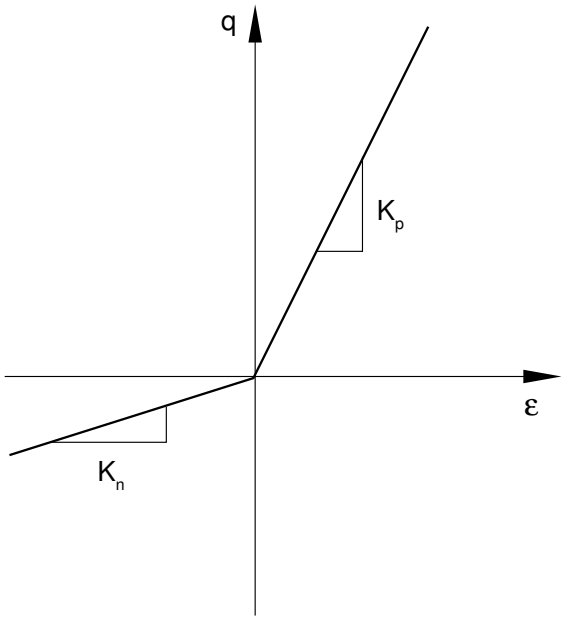


Figure 2: Linear constitutive model.

Input File Usage: **PIPE-SOIL STIFFNESS*, TYPE=LINEAR

Nonlinear Model

To define a nonlinear constitutive model, you specify the q_i relationship as a function of positive and negative relative displacement (“strain”), temperature, and field variables (see *Figure 3*). The behavior is assumed symmetric about the origin if only positive or negative data are provided.

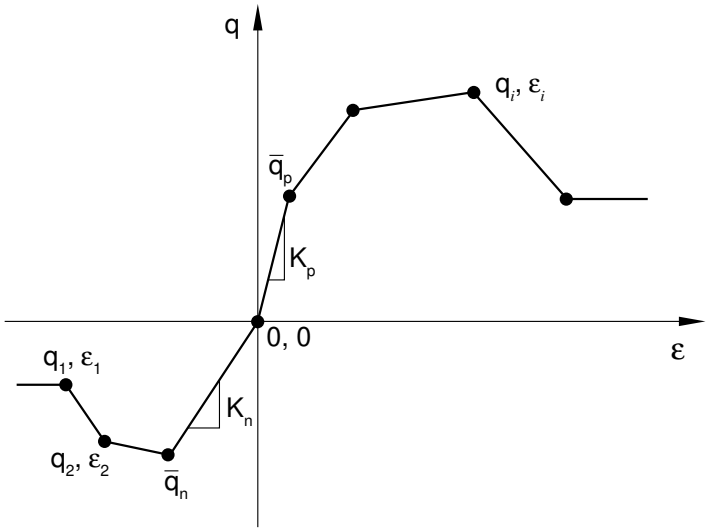


Figure 3: Nonlinear constitutive relationship.

You must provide the data in ascending order of relative displacement; you should provide it over a sufficiently wide range of relative displacement values so that the behavior is defined correctly. The force remains constant outside the range of data points. You must separate positive and negative data by specifying the data point at the origin of the force-relative displacement diagram. The two data points immediately before and after the data

point at the origin define the elastic stiffness, K_n and K_p, and the initial elastic limits, $\bar{q}_p$ and $\bar{q}_n$, as indicated in *Figure 3*.

The model provides linear elastic behavior if

$$F_n = q - \bar{q}_n\left(\bar{\varepsilon}_n^{pl}\right) \geq 0 \quad \text{and} \quad F_p = q - \bar{q}_p\left(\bar{\varepsilon}_p^{pl}\right) \leq 0,$$

where $\bar{\varepsilon}_n^{pl}$ and $\bar{\varepsilon}_p^{pl}$ are the equivalent plastic strains associated with negative and positive deformations, respectively. Inelastic deformation occurs when the relative force exceeds these elastic limits.

Hardening of the model is controlled by independent evolution of $\bar{q}_n\left(\bar{\varepsilon}_n^{pl}\right)$ and $\bar{q}_p\left(\bar{\varepsilon}_p^{pl}\right)$. The model assumes that $\bar{\varepsilon}_p^{pl}$ remains constant when the increment in relative displacement is negative, and $\bar{\varepsilon}_n^{pl}$ remains constant when the increment in relative displacement is positive. The response predicted by this model during a full loading cycle is shown in *Figure 4* for a simple constitutive law that uses different bilinear behavior associated with positive and negative force. *Figure 4* shows that the yield stress associated with positive force is updated to $\bar{q}_p$, while the initial yield stress associated with negative force, $\bar{q}_n^0$, remains constant during initial loading. Similarly, during subsequent reversed loading the yield stress associated with negative force is updated to $\bar{q}_n$, while the yield stress associated with positive force remains constant. Consequently, yielding occurs at $\bar{q}_p$ during the next load reversal. Such behavior is appropriate for the directions transverse to the pipeline where it is expected that relative positive motion between the pipe and soil is independent from relative negative motion between the pipe and soil.

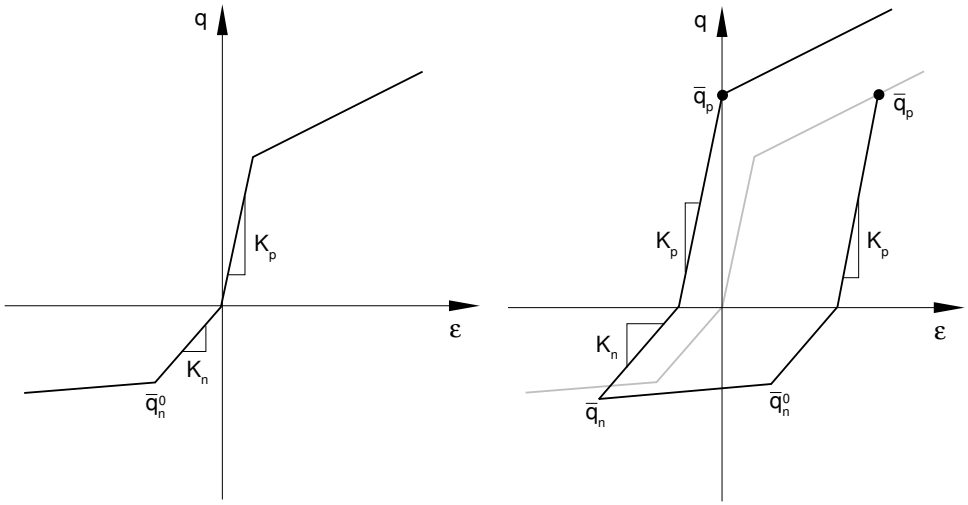


Figure 4: Cyclic loading for a bilinear model.

An isotropic hardening model is used if the behavior is symmetric about the origin (when only positive or negative data are provided). In this case only one equivalent plastic strain variable, $\bar{\varepsilon}^{pl}$, is used, which is updated when either negative or positive inelastic deformation occurs. Such an evolution model is more appropriate along the axial direction where it is expected that positive inelastic deformation influences subsequent negative inelastic deformation.

Input File Usage: **PIPE-SOIL STIFFNESS*, TYPE=NONLINEAR

Specifying the Constitutive Behavior Directly Using ASCE Formulae

Abaqus/Standard also provides analytical models to describe the pipe-soil interaction. These models define the constant ultimate force that can be exerted on the pipeline. In other words, these models describe elastic, perfectly plastic behavior. Forms of these formulae suitable for use with sands and clays are described in detail in the ASCE Guidelines for the Seismic Design of Oil and Gas Pipeline Systems.

The ASCE formulae can be applied in any arbitrary local system by associating an orientation definition with the element. However, these formulae are intended to be used in the default local coordinate system so that the formula for axial behavior is applied along the pipeline axis (the 1-direction), the formula for vertical behavior is applied along the 2-direction, and the formula for horizontal behavior along the 3-direction. You must specify the direction in which the behavior is specified when it is described by ASCE fomulae.

You specify all the parameters in the expressions below, except the depth, H, below the surface, which is measured along the edge of the PSI element as shown in *Figure 1* and is updated during geometrically nonlinear analysis. Values for the remaining parameters can be found in standard soil mechanics textbooks. Typical values are also provided in the ASCE Guidelines for the Seismic Design of Oil and Gas Pipeline Systems.

Axial Behavior

The ultimate axial load for sand, $\bar{q}_a$, is given by

$$\bar{q}_a = \tfrac{1}{2}\pi D\bar{\gamma}H\left(1+K_0\right)\tan\delta,$$

where K_0 is the coefficient of soil pressure at rest, H is the depth from the ground surface to the center of the pipeline, D is the external diameter of the pipeline, $\bar{\gamma}$ is the effective unit weight of soil, and δ is the interface angle of friction.

The ultimate axial load for clay is given by

$$\bar{q}_a = \pi D\alpha S,$$

where S is the undrained soil shear strength and α is an empirical adhesion factor that relates the undrained soil shear strength to the cohesion, $c = \alpha S$.

The maximum load is reached at an ultimate relative displacement, $\bar{\varepsilon}_a$, of approximately 2.5 to 5.0 mm (0.1 to 0.2 inches) for sand and approximately 2.5 to 10.0 mm (0.2 to 0.4 inches) for clay. A linear elastic response is assumed for $\varepsilon < \bar{\varepsilon}_a$.

The axial behavior is assumed to be symmetric about the origin. Consequently, only one equivalent plastic strain variable, $\bar{\varepsilon}_a^{pl}$, describes the evolution of the model. The equivalent plastic strain is updated when either negative or positive inelastic deformation occurs.

Input File Usage: Use one of the following options to define the axial behavior:

**PIPE-SOIL STIFFNESS*, DIRECTION=AXIAL, TYPE=SAND
**PIPE-SOIL STIFFNESS*, DIRECTION=AXIAL, TYPE=CLAY

Transverse Vertical Behavior

The vertical behavior is described by different relationships for “upward” motion (when the pipeline rises with respect to the ground surface) and “downward” motion. Downward motions give rise to positive relative displacements so that positive forces are applied to the pipeline. Similarly, upward motions give rise to negative relative displacements and pipeline forces.

The ultimate force for downward motion of the pipe in sand is given by

$$\bar{q}_{vp} = \bar{\gamma} H N_q D + \tfrac{1}{2} \gamma D^2 N_\gamma,$$

where N_q and N_γ are bearing capacity factors for vertical strip footings, vertically loaded in the downward direction, and γ is the total soil unit weight. Other parameters are defined in the previous section. The ultimate force for downward motion of the pipe in clay is given by

$$\bar{q}_{vp} = S N_c D,$$

where N_c is a bearing capacity factor. The ultimate force is reached at a relative displacement of approximately $\bar{\varepsilon}_{vp} = 0.1D$ to $\bar{\varepsilon}_{vp} = 0.15D$ for both sand and clay.

The ultimate force for upward motion of the pipe in sand is given by

$$\bar{q}_{vn} = \bar{\gamma} H N_{qv} D,$$

and for clay by

$$\bar{q}_{vn} = S N_{cv} D,$$

where N_{qv} and N_{cv} are vertical uplift factors.

The ultimate force is reached at a relative displacement of approximately $\bar{\varepsilon}_{vn} = 0.01H$ to $\bar{\varepsilon}_{vn} = 0.02H$ for sand and $\bar{\varepsilon}_{vn} = 0.1H$ to $\bar{\varepsilon}_{vn} = 0.2H$ for clay.

The transverse vertical behavior is non-symmetric about the origin. Consequently, two equivalent plastic strain variables—one associated with negative relative displacement, $\bar{\varepsilon}^{pl}_{vn}$, and the other with positive relative displacement, $\bar{\varepsilon}^{pl}_{vp}$—are used to describe the evolution of the model. The model assumes that $\bar{\varepsilon}^{pl}_{vp}$ remains constant when the increment in relative displacement is negative, and $\bar{\varepsilon}^{pl}_{vn}$ remains constant when the increment in relative displacement is positive.

Input File Usage: Use one of the following options to define the vertical behavior:

**PIPE-SOIL STIFFNESS*, DIRECTION=VERTICAL, TYPE=SAND
**PIPE-SOIL STIFFNESS*, DIRECTION=VERTICAL, TYPE=CLAY

Transverse Horizontal Behavior

The horizontal force-relative displacement relationship for sand is given by

$$\bar{q}_h = \bar{\gamma} H N_{qh} D,$$

and for clay by

$$\bar{q}_h = S N_{ch} D,$$

where N_{qh} and N_{ch} are horizontal bearing capacity factors. Other variables are defined in the previous sections.

The ultimate force is reached at a relative displacement of approximately $\bar{\varepsilon}_h = C_h (H + D/2)$, where C_h is between 0.07 to 0.1 for loose sand, between 0.03 to 0.05 for medium sand and clay, and between 0.02 to 0.03 for dense sand.

The transverse horizontal behavior is assumed to be symmetric about the origin. Consequently, only one equivalent plastic strain variable, $\bar{\varepsilon}^{pl}_h$, describes the evolution of the model. The equivalent plastic strain is updated when either negative or positive inelastic deformation occurs.

Input File Usage: Use one of the following options to define the horizontal behavior:

**PIPE-SOIL STIFFNESS*, DIRECTION=HORIZONTAL, TYPE=SAND
**PIPE-SOIL STIFFNESS*, DIRECTION=HORIZONTAL, TYPE=CLAY

Specifying the Directions for Which the Constitutive Behavior Is Defined

If you are defining the constitutive behavior by specifying the data directly, by default an isotropic model is assumed. If the model is not isotropic, you can specify different constitutive relationships in each direction. For two-dimensional nonisotropic models you must specify the behavior in two directions; for three-dimensional nonisotropic models you must specify the behavior in three directions. You must indicate the direction in which the behavior is specified. You can specify the 1-direction, 2-direction, 3-direction, axial direction, vertical direction, or horizontal direction. Abaqus/Standard assumes that the axial direction is equivalent to the 1-direction, the vertical direction is equivalent to the 2-direction, and the horizontal direction is equivalent to the 3-direction.

Input File Usage: Use the following option to define an isotropic constitutive model:

**PIPE-SOIL STIFFNESS*

Use the following option to define the constitutive model in a particular direction:

**PIPE-SOIL STIFFNESS*, DIRECTION=*direction*

where *direction* can be 1, 2, 3, AXIAL, VERTICAL, or HORIZONTAL. Repeat the **PIPE-SOIL STIFFNESS* option with the DIRECTION parameter as many times as necessary to define the behavior in each direction.

Output

The force per unit length in the element local system is available through the “stress” output variable S. Relative deformation is available through the “strain” output variable E. Elastic and plastic “strains” are available through the output variables EE and PE.

Element nodal force (the force the element places on the pipeline nodes, in the global system) is available through element variable NFORC.

References

- Audibert, J. M. E., D. J. Nyman, and T. D. O'Rourke, “Differential Ground Movement Effects on Buried Pipelines,” *Guidelines for the Seismic Design of Oil and Gas Pipeline Systems, ASCE publication*, pp. 151–180, 1984.

Pipe-Soil Interaction Element Library

Products: Abaqus/Standard

Overview

References:

- *Pipe-Soil Interaction Elements*
- **PIPE-SOIL INTERACTION*

Element Types

2D Elements

PSI24	Two-dimensional 4-node pipe-soil interaction element
PSI26	Two-dimensional 6-node pipe-soil interaction element

Active Degrees of Freedom

1, 2

Additional Solution Variables

None.

3D Elements

PSI34	Three-dimensional 4-node pipe-soil interaction element
PSI36	Three-dimensional 6-node pipe-soil interaction element

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

None.

Nodal Coordinates Required

2D: *X*, *Y*

3D: *X*, *Y*, *Z*

Element Property Definition

Input File Usage: **PIPE-SOIL INTERACTION*

Element-Based Loading

None.

Element Output

The relative displacements corresponding to the forces below are chosen by requesting the corresponding "strains." Elastic and plastic strains are available.

Two-Dimensional Elements

S11 Force per unit length in the first local direction.

S22 Force per unit length in the second local direction.

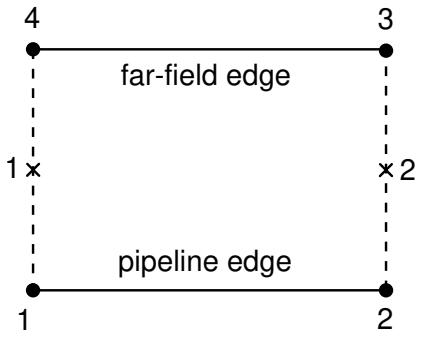
Three-Dimensional Elements

S11 Force per unit length in the first local direction.

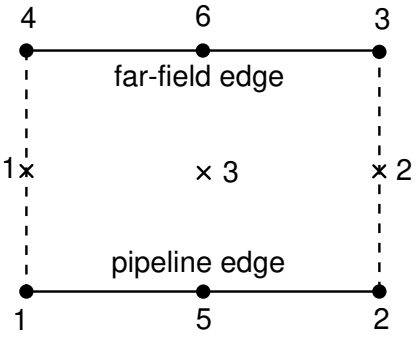
S22 Force per unit length in the second local direction.

S33 Force per unit length in the third local direction.

Node Ordering and Integration Point Numbering



PSI24 and PSI34



PSI26 and PSI36

Spring Elements

In this section:

- *Springs*
- *Spring Element Library*

Springs

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Spring Element Library*
- **SPRING*
- *Defining springs and dashpots*

Overview

Spring elements:

- can couple a force with a relative displacement;
- in Abaqus/Standard can couple a moment with a relative rotation;
- can be linear or nonlinear;
- if linear, can be dependent on frequency in direct-solution steady-state dynamic analysis;
- can be dependent on temperature and field variables; and
- can be used to assign a structural damping factor to form the imaginary part of spring stiffness.

The terms “force” and “displacement” are used throughout the description of spring elements. When the spring is associated with displacement degrees of freedom, these variables are the force and relative displacement in the spring. If the springs are associated with rotational degrees of freedom, they are torsional springs; these variables will then be the moment transmitted by the spring and the relative rotation across the spring.

Viscoelastic spring behavior can be modeled in Abaqus/Standard by combining frequency-dependent springs and frequency-dependent dashpots.

Typical Applications

Spring elements are used to model actual physical springs as well as idealizations of axial or torsional components. They can also model restraints to prevent rigid body motion.

They are also used to represent structural dampers by specifying structural damping factors to form the imaginary part of the spring stiffness.

Choosing an Appropriate Element

SPRING1 and SPRING2 elements are available only in Abaqus/Standard. SPRING1 is between a node and ground, acting in a fixed direction. SPRING2 is between two nodes, acting in a fixed direction.

The SPRINGA element is available in both Abaqus/Standard and Abaqus/Explicit. SPRINGA acts between two nodes, with its line of action being the line joining the two nodes, so that this line of action can rotate in large-displacement analysis.

The spring behavior can be linear or nonlinear in any of the spring elements in Abaqus.

Element types SPRING1 and SPRING2 can be associated with displacement or rotational degrees of freedom (in the latter case, as torsional springs). However, the use of torsional springs in large-displacement analysis requires careful consideration of the definition of total rotation at a node; therefore, connector elements (*About Connectors*) are usually a better approach to providing torsional springs for large-displacement cases.

Input File Usage: Use the following option to specify a spring element between a node and ground, acting in a fixed direction:

**ELEMENT*, TYPE=SPRING1

Use the following option to specify a spring element between two nodes, acting in a fixed direction:

**ELEMENT*, TYPE=SPRING2

Use the following option to specify a spring element between two nodes with its line of action being the line joining the two nodes:

**ELEMENT*, TYPE=SPRINGA

Abaqus/CAE Usage: Property or Interaction module: **Special->Springs/Dashpots->Create**, then select one of the following:
Connect points to ground: select points: toggle on **Spring stiffness**
(*equivalent to SPRING1*)
Connect two points: select points: **Axis**: **Specify fixed direction**:
toggle on **Spring stiffness**
(*equivalent to SPRING2*)
Connect two points: select points: **Axis**: **Follow line of action**:
toggle on **Spring stiffness**
(*equivalent to SPRINGA*)

Stability Considerations in Abaqus/Explicit

A SPRINGA element introduces a stiffness between two degrees of freedom without introducing an associated mass. In an explicit dynamic procedure this represents an unconditionally unstable element. The nodes to which the spring is attached must have some mass contribution from adjacent elements; if this condition is not satisfied, Abaqus/Explicit will issue an error message. If the spring is not too stiff (relative to the stiffness of the adjacent elements), the stable time increment determined by the explicit dynamics procedure (*Explicit Dynamic Analysis*) will suffice to ensure stability of the calculations.

Abaqus/Explicit does not use the springs in the determination of the stable time increment. During the data check phase of the analysis, Abaqus/Explicit computes the minimum of the stable time increment for all the elements in the mesh except the spring elements. The program then uses this minimum stable time increment and the stiffness of each of the springs to determine the mass required for each spring to give the same stable time increment. If this mass is too large compared to the mass of the model, Abaqus/Explicit will issue an error message that the spring is too stiff compared to the model definition.

Relative Displacement Definition

The relative displacement definition depends on the element type.

SPRING1 Elements

The relative displacement across a SPRING1 element is the *i*th component of displacement of the spring's node:

$$\Delta u = u_i,$$

where i is defined as described below and can be in a local direction (see *Defining the Direction of Action for SPRING1 and SPRING2 Elements*).

SPRING2 Elements

The relative displacement across a SPRING2 element is the difference between the *i*th component of displacement of the spring's first node and the *j*th component of displacement of the spring's second node:

$$\Delta u = u_i^1 - u_j^2,$$

where *i* and *j* are defined as described below and can be in local directions (see *Defining the Direction of Action for SPRING1 and SPRING2 Elements*).

It is important to understand how the SPRING2 element will behave according to the above relative displacement equation since the element can produce counterintuitive results. For example, a SPRING2 element set up in the following way will be a "compressive" spring:



If the nodes displace so that $u_i^1 = 1$ and $u_j^2 = 0$, the spring appears to be in compression, while the force in the SPRING2 element is positive. To obtain a "tensile" spring, the SPRING2 element should be set up in the following way:



SPRINGA Elements

For geometrically linear analysis the relative displacement is measured along the direction of the SPRINGA element in the reference configuration:

$$\Delta u = \frac{(\mathbf{x}^1 - \mathbf{x}^2)}{\sqrt{(\mathbf{x}^1 - \mathbf{x}^2) \cdot (\mathbf{x}^1 - \mathbf{x}^2)}} \cdot \left(\mathbf{u}^1 - \mathbf{u}^2\right),$$

where $\mathbf{X}^1$ is the reference position of the first node of the spring and $\mathbf{X}^2$ is the reference position of its second node.

For geometrically nonlinear analysis the relative displacement across a SPRINGA element is the change in length in the spring between the initial and the current configuration:

$$\Delta u = l - l_0,$$

where $l = \sqrt{(\mathbf{x}^1 - \mathbf{x}^2) \cdot (\mathbf{x}^1 - \mathbf{x}^2)}$ is the current length of the spring and l_0 is the value of l in the initial configuration. Here $\mathbf{x}^1$ and $\mathbf{x}^2$ are the current positions of the nodes of the spring.

In either case the force in a SPRINGA element is positive in tension.

Defining Spring Behavior

The spring behavior can be linear or nonlinear. In either case you must associate the spring behavior with a region of your model.

Input File Usage: **SPRING*, ELSET=*name*

where the ELSET parameter refers to a set of spring elements.

Abaqus/CAE Usage: Property or Interaction module: **Special->Springs/Dashpots->Create**: select connectivity type: select points

Defining Linear Spring Behavior

You define linear spring behavior by specifying a constant spring stiffness (force per relative displacement).

The spring stiffness can depend on temperature and field variables. See *Input Syntax Rules* for further information about defining data as functions of temperature and independent field variables.

For direct-solution steady-state dynamic analysis the spring stiffness can depend on frequency, as well as on temperature and field variables. If a frequency-dependent spring stiffness is specified for any other analysis procedure in Abaqus/Standard, the data for the lowest frequency given will be used.

Input File Usage: **SPRING*, DEPENDENCIES=*n*
first data line
spring stiffness, *frequency*, *temperature*, *field variable 1*, etc.
...

Abaqus/CAE Usage: Property or Interaction module: **Special->Springs/Dashpots->Create**: select connectivity type: select points: **Property**: **Spring stiffness**:
spring stiffness

Defining the spring stiffness as a function of frequency, temperature, and field variables is not supported in Abaqus/CAE when you define springs as engineering features; instead, you can define connectors that have spring-like elastic behavior (see *Connector Elastic Behavior*).

Defining Nonlinear Spring Behavior

You define nonlinear spring behavior by giving pairs of force–relative displacement values. These values should be given in ascending order of relative displacement and should be provided over a sufficiently wide range of relative displacement values so that the behavior is defined correctly. Abaqus assumes that the force remains constant (which results in zero stiffness) outside the range given (see *Figure 1*).

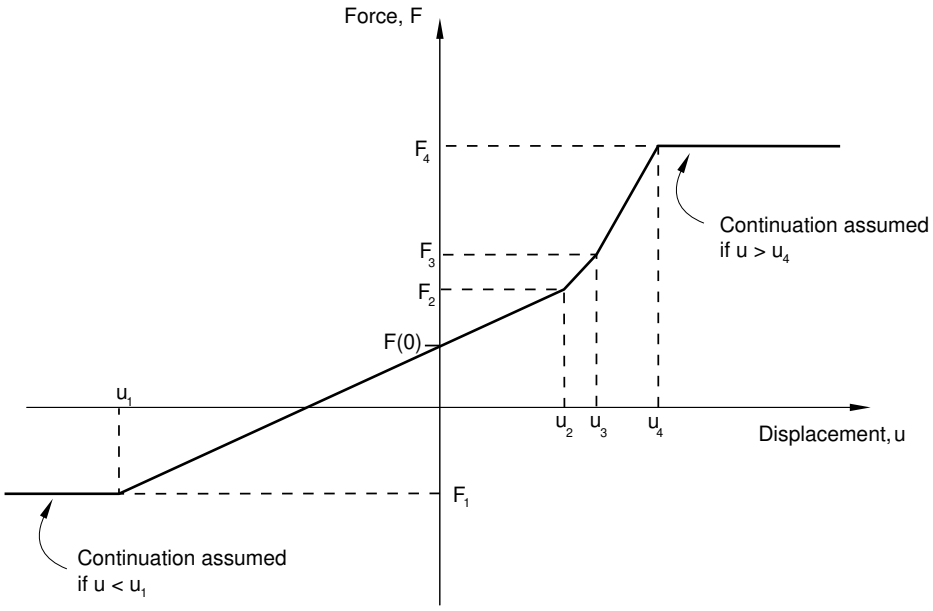


Figure 1: Nonlinear spring force–relative displacement relationship.

Initial forces in nonlinear springs should be defined as part of the $\boldsymbol{F(u)}$ relationship by giving a nonzero force, $\boldsymbol{F(0)}$, at zero relative displacement.

The spring stiffness can depend on temperature and field variables. See *Input Syntax Rules* for further information about defining data as functions of temperature and independent field variables.

Abaqus/Explicit will regularize the data into tables that are defined in terms of even intervals of the independent variables. In some cases where the force is defined at uneven intervals of the independent variable (relative displacement) and the range of the independent variable is large compared to the smallest interval, Abaqus/Explicit may fail to obtain an accurate regularization of your data in a reasonable number of intervals. In this case the program will stop after all data are processed with an error message that you must redefine the material data. See *Material Data Definition* for a more detailed discussion of data regularization.

Input File Usage: **SPRING*, NONLINEAR, DEPENDENCIES=*n*
first data line
force, *relative displacement*, *temperature*, *field variable 1*, etc.
...

Abaqus/CAE Usage: Defining nonlinear spring behavior is not supported in Abaqus/CAE when you define springs as engineering features; instead, you can define connectors that have spring-like elastic behavior (see *Connector Elastic Behavior*).

Defining the Direction of Action for SPRING1 and SPRING2 Elements

You define the direction of action for SPRING1 and SPRING2 elements by giving the degree of freedom at each node of the element. This degree of freedom may be in a local coordinate system (*Orientations*). The local system is assumed to be fixed: even in large-displacement analysis SPRING1 and SPRING2 elements act in a fixed direction throughout the analysis.

Input File Usage: **SPRING*, ORIENTATION=*name*
dof at node 1, *dof at node 2*

Abaqus/CAE Usage: Property or Interaction module: **Special->Springs/Dashpots->Create**, then select one of the following:
Connect points to ground: select points: **Orientation**: **Edit**:
select orientation
Connect two points: select points: **Axis**: **Specify fixed direction**: **Orientation**:
Edit: select orientation

Defining Linear Spring Behavior with Complex Stiffness

Springs can be used to simulate structural dampers that contribute to the imaginary part of the element stiffness forming an elemental structural damping matrix. You specify both the real part of the spring stiffness for particular degrees of freedom and the structural damping factor, s. The imaginary part of the spring stiffness is calculated as isK and represents structural damping. These data can be frequency dependent.

Input File Usage: **SPRING*, COMPLEX STIFFNESS
first data line
real spring stiffness, *structural damping factor*, *frequency*

Abaqus/CAE Usage: Linear spring behavior with complex stiffness is not supported in Abaqus/CAE.

Spring Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

Overview

This section provides a reference to the spring elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *Springs*
- **SPRING*

Element Types

SPRINGA	Axial spring between two nodes, whose line of action is the line joining the two nodes. This line of action may rotate in large-displacement analysis
SPRING1$^{(S)}$	Spring between a node and ground, acting in a fixed direction
SPRING2$^{(S)}$	Spring between two nodes, acting in a fixed direction

Active Degrees of Freedom

SPRINGA: 1, 2, 3. The translational degree of freedom in the 3-direction is not activated in an Abaqus/Standard analysis if both nodes of the element are connected to two-dimensional entities such as two-dimensional analytical rigid surfaces, two-dimensional beam elements, etc.

SPRING1 or SPRING2: 1, 2, 3, 4, 5, or 6. If you specify a local orientation for the spring, these are local degrees of freedom. Otherwise, these are global degrees of freedom.

Additional Solution Variables

None.

Nodal Coordinates Required

SPRINGA: *X*, *Y*, *Z*. These coordinates are used in the calculation of the action of the element.

SPRING1 or SPRING2: None. The element nodes do not need to have coordinates defined since the action associated with these elements is defined by specifying the degrees of freedom involved.

Element Property Definition

Input File Usage: **SPRING*

Abaqus/CAE Usage: Property or Interaction module: **Special->Springs/Dashpots->Create**

Element-Based Loading

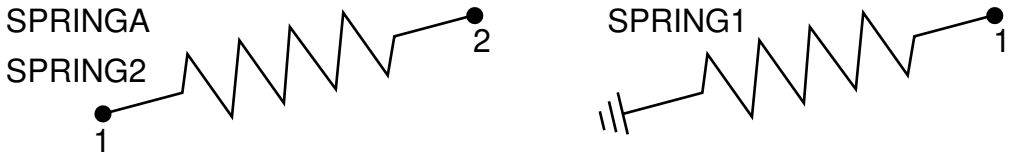
None.

Element Output

S11 Force in the spring.

E11 Relative displacement across the spring.

Node Ordering on Elements



Surface Elements

In this section:

- *Surface Elements*
- *General Surface Element Library*
- *Cylindrical Surface Element Library*
- *Axisymmetric Surface Element Library*

Surface Elements

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE Abaqus/Aqua

References:

- *General Surface Element Library*
- *Cylindrical Surface Element Library*
- *Axisymmetric Surface Element Library*
- **SURFACE SECTION*
- *Creating surface sections*

Overview

Surface elements:

- are defined just like membrane elements—as surfaces in space;
- have no inherent stiffness;
- can have mass per unit area;
- can be used to define rigid bodies;
- can be used in the definition of surfaces and surface-based tie constraints;
- behave just like membrane elements with zero thickness;
- can be defined with thickness (only affects contact);
- can be used with rebar layers;
- can be embedded in solid elements;
- can transmit only in-plane forces; and
- have no bending stiffness or transverse shear stiffness.

Typical Applications

Surface elements are useful in several special modeling cases:

- They are used to carry rebar layers to represent thin stiffening components in solid structures. The stiffness and mass of the rebar layers are added to the surface elements (see *Defining Reinforcement*). The reinforced surface elements can also be embedded in "host" solid elements (see *Embedded Elements*).
- They are used to bring additional mass into the model in the form of a mass per unit area; for example, to spread the mass of fuel in a tank over the tank surface, particularly when the tank is modeled with solid elements.
- They are used to specify a surface used in a constraint, when that surface does not have structural properties.
- When used in conjunction with a surface-based tie constraint, they are used to specify distributed surface loading, such as incident wave loading, on beam elements.
- In Abaqus/Explicit (when used in conjunction with a surface-based tie constraint) they can be used to specify a complex surface on beam elements for use in general contact. The stiffness of the penalty springs used to enforce contact constraints is approximately proportional to the mass of the surface nodes. Contact will not be enforced if the surface nodes have no mass.

- In Abaqus/Explicit they can be used to define all or part of the boundary for a surface-based fluid cavity (for example, see *Hydrostatic fluid elements: modeling an airspring*).
- In Abaqus/Aqua analysis they can be used to visualize gravity waves.

Choosing an Appropriate Element

In addition to the general surface elements available in both Abaqus/Standard and Abaqus/Explicit, cylindrical surface elements and axisymmetric surface elements are available in Abaqus/Standard only.

General Surface Elements

General surface elements should be used in three-dimensional models in which the deformation of the structure can evolve in three dimensions.

Cylindrical Surface Elements

Cylindrical surface elements are available in Abaqus/Standard for precise modeling of regions in a structure with circular geometry, such as a tire. The elements make use of trigonometric functions to interpolate displacements along the circumferential direction and use regular isoparametric interpolation in the in-plane direction. They use three nodes along the circumferential direction and can span a segment between 0° and 180°. Elements with both first-order and second-order interpolation in the in-plane direction are available.

The geometry of the element is defined by specifying nodal coordinates in a global Cartesian system.

These elements can be used in the same mesh with regular surface elements. They can also be embedded in general solid and cylindrical elements.

Axisymmetric Surface Elements

The axisymmetric surface elements available in Abaqus/Standard are divided into two categories: those that do not allow twist about the symmetry axis and those that do. These elements are referred to as the regular and generalized axisymmetric surface elements, respectively.

The generalized axisymmetric surface elements (axisymmetric surface elements with twist) allow a circumferential component of loading, which might cause twist about the symmetry axis. The circumferential load component is independent of the circumferential coordinate θ. Since there is no dependence of the loading on the circumferential coordinate, the deformation is axisymmetric.

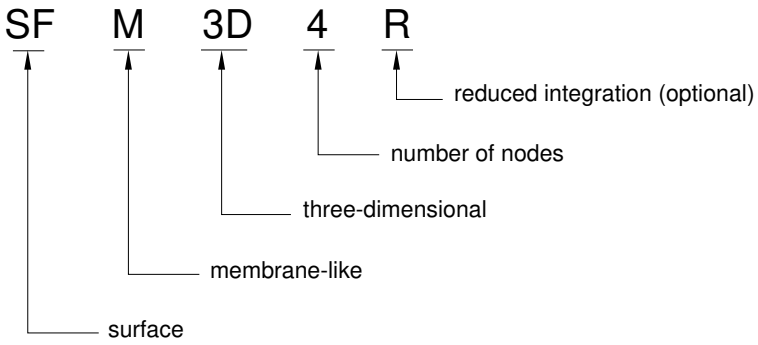
The generalized axisymmetric surface elements cannot be used in dynamic or eigenfrequency extraction procedures.

Naming Convention

The naming convention for surface elements depends on the element dimensionality.

General Surface Elements

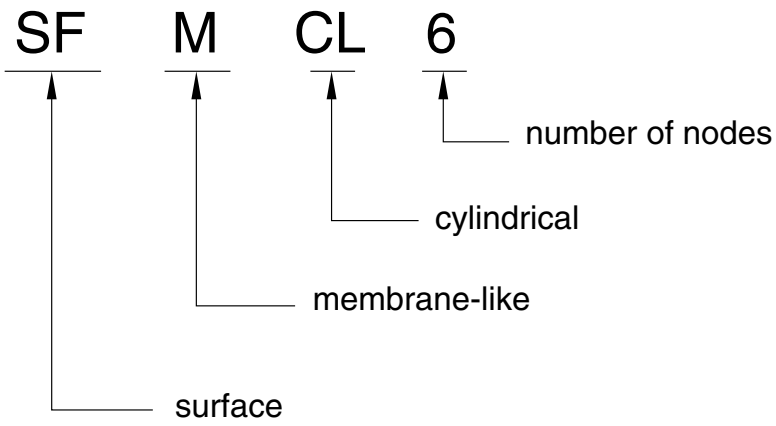
General surface elements in Abaqus are named as follows:



For example, SFM3D4R is a three-dimensional, 4-node surface element with reduced integration.

Cylindrical Surface Elements

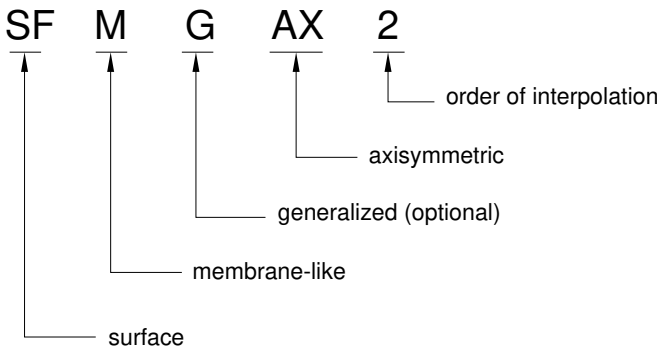
Cylindrical surface elements in Abaqus/Standard are named as follows:



For example, SFMCL6 is a 6-node cylindrical surface element with circumferential interpolation.

Axisymmetric Surface Elements

Axisymmetric surface elements in Abaqus/Standard are named as follows:



For example, SFMAX2 is a regular axisymmetric, quadratic-interpolation surface element.

Element Normal Definition

The “top” surface of a surface element is the surface in the positive normal direction (defined below) and is called the SPOS face for contact definition. The “bottom” surface is in the negative direction along the normal and is called the SNEG face for contact definition.

General Surface Elements

For general surface elements the positive normal direction is defined by the right-hand rule going around the nodes of the element in the order that they are specified in the element definition. See *Figure 1*.

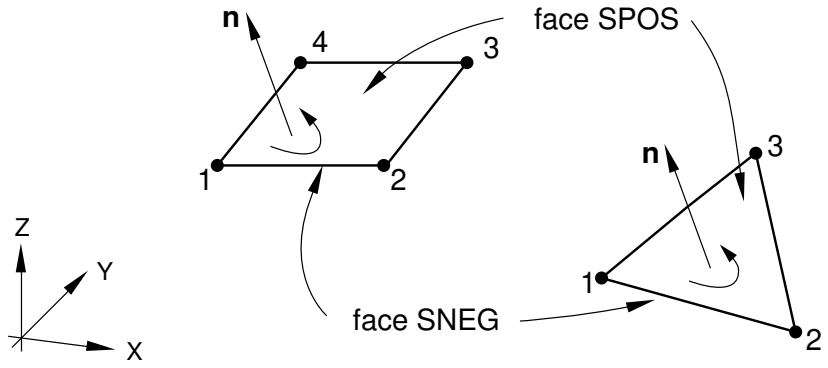


Figure 1: Positive normals for general surface elements.

Cylindrical Surface Elements

The positive normal direction is defined by the right-hand rule going around the nodes of the element in the order that they are specified in the element definition. See *Figure 2*.

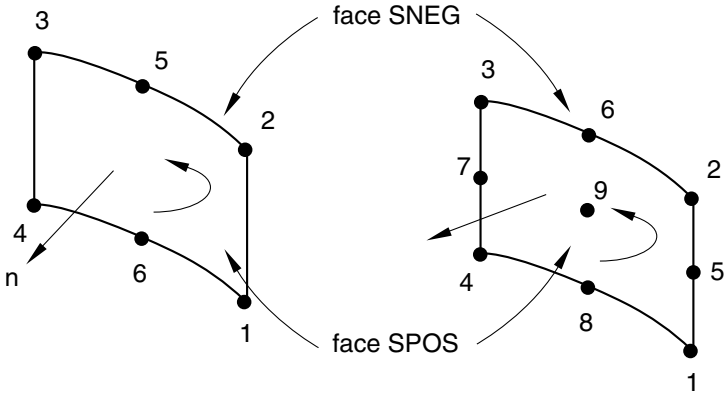


Figure 2: Positive normals for cylindrical surface elements.

Axisymmetric Surface Elements

For axisymmetric surface elements the positive normal is defined by a 90° counterclockwise rotation from the direction going from node 1 to node 2. See *Figure 3*.

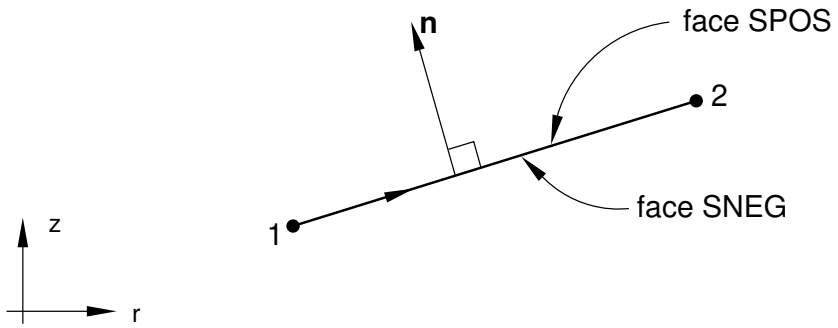


Figure 3: Positive normals for axisymmetric surface elements.

Defining the Elements Section Properties

You must associate the surface section properties with a region of your model.

Input File Usage: **SURFACE SECTION*, ELSET=*name*

where the ELSET parameter refers to a set of surface elements.

Abaqus/CAE Usage: Property module:
Create Section: select **Shell** as the section **Category** and **Surface** as the section **Type**
Assign->Section: select regions

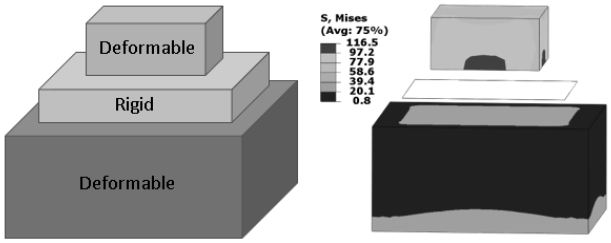
Defining a Constant Section Thickness

You can define a constant section thickness as part of the section definition.

By default, user-defined surface elements have zero element thickness. Internally generated surface elements have nonzero element thickness in some cases. The element thickness results in a contact thickness for the surface elements within the contact domain.

Element calculations (including rebar layers for defining reinforcements or mass and rotary inertia properties, for example) are computed as if the section thickness is zero. Therefore, any specified section thickness only affects contact calculations.

A surface element with thickness that is assigned to a rigid body can be used to model a rigid part, such as shown below. A pressure load is applied to the top surface of the green deformable body in the figure. The bottom of the blue deformable body is held fixed. Density per unit area and thickness corresponding to the original distance between the two deformable bodies is assigned in the section definition of the surface element. The image on the right shows the Mises stress results from the simulation.



Input File Usage: **SURFACE SECTION*, ELSET=*name*
thickness

Using a Surface Element to Carry Rebar Layers

You can define layers of reinforcement that are carried by the surface element. The stiffness and mass due to the rebar layers are added to the surface element.

Input File Usage: Use both of the following options:

**SURFACE SECTION*, ELSET=*name*
**REBAR LAYER*

Abaqus/CAE Usage: Property module: **Create Section**: select **Shell** as the section **Category** and **Surface** as the section **Type**, **Rebar Layers**

Using a Surface Element to Bring Additional Mass into the Model

You can define the mass per unit area carried by the surface element.

Input File Usage: **SURFACE SECTION*, ELSET=*name*, DENSITY=*number*

Abaqus/CAE Usage: Property module: **Create Section**: select **Shell** as the section **Category** and **Surface** as the section **Type**, toggle on **Density**: *number*

Using a Surface Element in a Constraint

Surface elements can be used to define a surface in Abaqus, and this surface can be used in a surface-based tie constraint (see *Mesh Tie Constraints*).

Input File Usage: Use the following options:

**SURFACE*, NAME=*surface_name*
**TIE*, NAME=*name*
surface_name, *main_name*

Abaqus/CAE Usage: In Abaqus/CAE you can select one or more faces directly in the viewport when you are prompted to select a surface. In addition, you can define surfaces as collections of faces and edges using the Surface toolset.

Interaction module: **Create Constraint**: **Tie**

Using a Surface Element to Visualize Gravity Waves

You can define a surface element set at the still water height to visualize the gravity waves during an Abaqus/Aqua analysis.

Input File Usage: **SURFACE SECTION*, ELSET=*name*, AQUAVISUALIZATION=YES

Abaqus/CAE Usage: Specifying a wave surface for visualization is not supported in Abaqus/CAE.

General Surface Element Library

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE Abaqus/Aqua

Overview

This section provides a reference to the surface elements available in Abaqus/Standard, Abaqus/Explicit, and Abaqus/Aqua.

References:

- *Surface Elements*
- **SURFACE SECTION*
- **REBAR LAYER*

Element Types

SFM3D3	3-node triangle
SFM3D4$^{(S)}$	4-node quadrilateral
SFM3D4R	4-node quadrilateral, reduced integration
SFM3D6$^{(S)}$	6-node triangle
SFM3D8$^{(S)}$	8-node quadrilateral
SFM3D8R$^{(S)}$	8-node quadrilateral, reduced integration

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage: Use the following option to define surface element properties:

**SURFACE SECTION*

If rebar are being defined, use the following option in conjunction with the **SURFACE SECTION* option:

**REBAR LAYER*

Use the following option to define a mass density per unit area:

**SURFACE SECTION*, DENSITY=*number*

Use the following option to define the free surface of water in an Abaqus/Aqua analysis:

**SURFACE SECTION*, AQUAVISUALIZATION=YES

Abaqus/CAE Usage: Property module: **Create Section**: select **Shell** as the section **Category** and **Surface** as the section **Type**, **Rebar Layers** (optional)

You cannot define the mass per unit area or the free surface of water for a surface section in Abaqus/CAE.

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*. Gravity, centrifugal, rotary acceleration, and Coriolis force loads apply only if the surface elements have rebar defined or if the elements have a defined density.

Load ID (*DLOAD): BX
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-2}
Description: Body force in the global *X*-direction.

Load ID (*DLOAD): BY
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-2}
Description: Body force in the global *Y*-direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-2}
Description: Body force in the global *Z*-direction.

Load ID (*DLOAD): BXNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-2}
Description: Nonuniform body force in the global *X*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BYNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-2}

Description: Nonuniform body force in the global *Y*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-2}
Description: Nonuniform body force in the global *Z*-direction with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit.

Load ID (*DLOAD): CENT (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-3} ($ML^{-2}T^{-2}$)

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit area, ω is the angular speed).

Load ID (*DLOAD): CENTRIF (S)
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular speed).

Load ID (*DLOAD): CORIO (S)
Abaqus/CAE Load/Interaction: Coriolis force
Units: $FL^{-3}T$ ($ML^{-2}T^{-1}$)

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit area, ω is the angular speed). The load stiffness due to Coriolis loading is not accounted for in direct steady-state dynamics analysis.

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): HP (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Hydrostatic pressure applied to the element reference surface and linear in global *Z*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure applied to the element reference surface. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): PNU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure applied to the element reference surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): ROTA (S)

Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Rotary acceleration load (magnitude is input as $\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is the rotary acceleration).

Load ID (*DLOAD): SBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-5}T^2$

Description: Stagnation body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): SP (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T^2$

Description: Stagnation pressure applied to the element reference surface.

Load ID (*DLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Shear traction on the element reference surface.

Load ID (*DLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: General traction on the element reference surface.

Load ID (*DLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}

Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): VBF (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-4}T$

Description: Viscous body force in global *X*-, *Y*-, and *Z*-directions.

Load ID (*DLOAD): VP (E)
Abaqus/CAE Load/Interaction: Not supported
Units: $FL^{-3}T$

Description: Viscous surface pressure applied to the element reference surface. The pressure is proportional to the velocity normal to the element face and opposing the motion.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): HP (S)
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Hydrostatic pressure on the element reference surface and linear in global *Z*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure on the element reference surface. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Nonuniform pressure on the element reference surface with magnitude supplied via user subroutine *DLOAD* in Abaqus/Standard and *VDLOAD* in Abaqus/Explicit. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): SP (E)
Abaqus/CAE Load/Interaction: Pressure
Units: $FL^{-4}T^2$
Description: Stagnation pressure applied to the element reference surface.

Load ID (*DSLOAD): ST
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Surface tension applied to the element reference surface.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DSLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element reference surface.

Load ID (*DSLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): VP (E)
Abaqus/CAE Load/Interaction: Pressure

Units: $FL^{-3}T$

Description: Viscous surface pressure applied to the element reference surface. The pressure is proportional to the velocity normal to the element surface and opposing the motion.

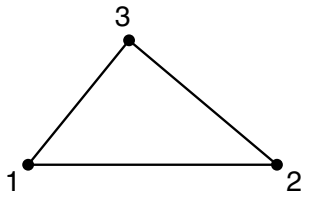
Incident Wave Loading

Surface-based incident wave loading is also available for these elements. See *Acoustic and Shock Loads*.

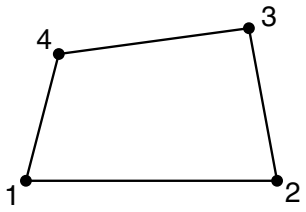
Element Output

Output is currently available only when the surface element is used to carry rebar layers. See *Defining Reinforcement* for details.

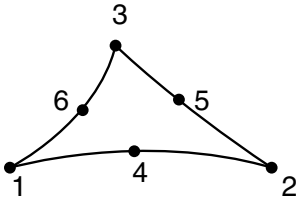
Node Ordering on Elements



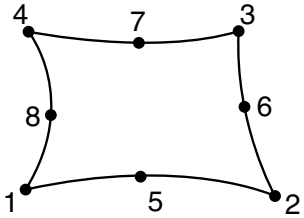
3 - node element



4 - node element

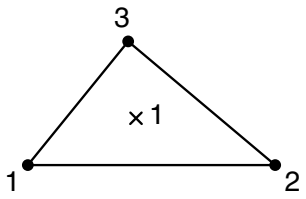


6 - node element

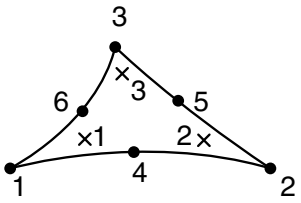


8 - node element

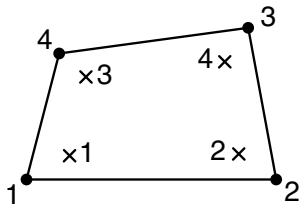
Numbering of Integration Points for Output



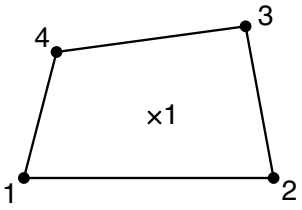
3 - node element



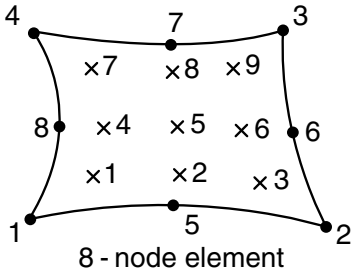
6 - node element



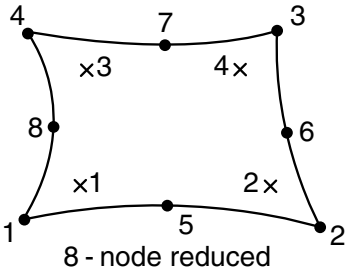
4 - node element



4 - node reduced integration element



8 - node element



8 - node reduced integration element

Cylindrical Surface Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the cylindrical surface elements available in Abaqus/Standard.

References:

- *Surface Elements*
- **SURFACE SECTION*
- **REBAR LAYER*

Element Types

SFMCL6	6-node cylindrical surface
SFMCL9	9-node cylindrical surface

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage: Use the following option to define surface element properties:

**SURFACE SECTION*

If rebar are being defined, use the following option in conjunction with the **SURFACE SECTION* option:

**REBAR LAYER*

Use the following option to define a mass density per unit area:

**SURFACE SECTION*, DENSITY=*number*

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*. Gravity, centrifugal, rotary acceleration, and Coriolis force loads apply only if the surface elements have rebar defined or if the elements have a defined density.

Load ID (*DLOAD): BX
Units: FL^{-3}
Description: Body force in the global *X*-direction.

Load ID (*DLOAD): BY
Units: FL^{-2}
Description: Body force in the global *Y*-direction.

Load ID (*DLOAD): BZ
Units: FL^{-2}
Description: Body force in the global *Z*-direction.

Load ID (*DLOAD): BXNU
Units: FL^{-2}
Description: Nonuniform body force in the global *X*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BYNU
Units: FL^{-2}
Description: Nonuniform body force in the global *Y*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BZNU
Units: FL^{-2}
Description: Nonuniform body force in the global *Z*-direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): CENT
Units: $FL^{-3}(ML^{-2}\,T^{-2})$

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit area, ω is the angular speed).

Load ID (*DLOAD): CENTRIF
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular speed).

Load ID (*DLOAD): CORIO
Units: $FL^{-3}T\ (ML^{-2}\,T^{-1})$

Description: Coriolis force (magnitude is input as $\rho\omega$, where ρ is the mass density per unit area, ω is the angular speed). The load stiffness due to Coriolis loading is not accounted for in direct steady-state dynamics analysis.

Load ID (*DLOAD): GRAV
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude is input as acceleration).

Load ID (*DLOAD): HP
Units: FL^{-2}
Description: Hydrostatic pressure applied to the element reference surface and linear in global *Z*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): P
Units: FL^{-2}
Description: Pressure applied to the element reference surface. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): PNU
Units: FL^{-2}
Description: Nonuniform pressure applied to the element reference surface with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): ROTA
Units: T^{-2}
Description: Rotary acceleration load (magnitude is input as α, where α is the rotary acceleration).

Load ID (*DLOAD): TRSHR
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DLOAD): TRSHRNU (S)
Units: FL^{-2}
Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC
Units: FL^{-2}
Description: General traction on the element reference surface.

Load ID (*DLOAD): TRVECNU (S)
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): HP
Units: FL^{-2}
Description: Hydrostatic pressure on the element reference surface and linear in global *Z*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): P
Units: FL^{-2}
Description: Pressure on the element reference surface. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): PNU

Units: FL^{-2}

Description: Nonuniform pressure on the element reference surface with magnitude supplied via user subroutine *DLOAD*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): TRSHR
Units: FL^{-2}

Description: Shear traction on the element reference surface.

Load ID (*DSLOAD): TRSHRNU (S)
Units: FL^{-2}

Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Units: FL^{-2}

Description: General traction on the element reference surface.

Load ID (*DSLOAD): TRVECNU (S)
Units: FL^{-2}

Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

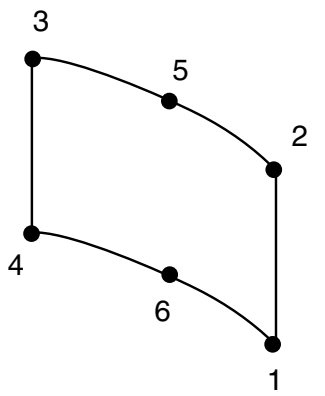
Incident Wave Loading

Surface-based incident wave loading is also available for these elements. See *Acoustic and Shock Loads*.

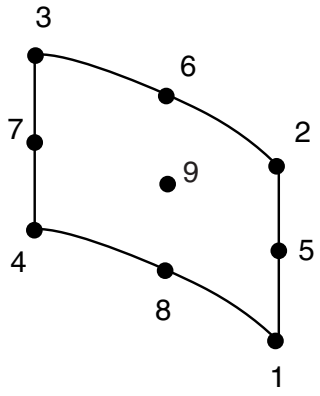
Element Output

Output is currently available only when the surface element is used to carry rebar layers. See *Defining Reinforcement* for details.

Node Ordering and Face Numbering on Elements

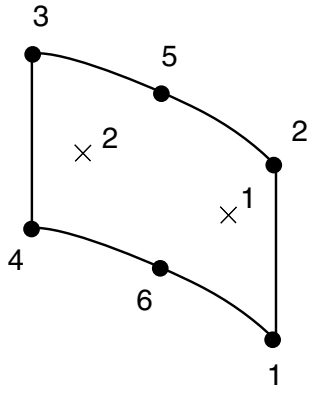


6-node element

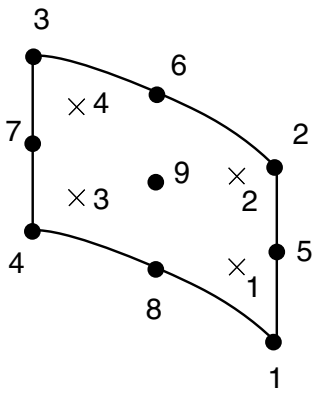


9-node element

Numbering of Integration Points for Output



6-node element



9-node element

Axisymmetric Surface Element Library

Products: Abaqus/Standard Abaqus/CAE

Overview

This section provides a reference to the axisymmetric surface elements available in Abaqus/Standard.

Conventions

Coordinate 1 is r, coordinate 2 is z. At $\theta = 0$, the r-direction corresponds to the global X-direction and the z-direction corresponds to the global Y-direction. This is important when data must be given in global directions. Coordinate 1 should be greater than or equal to zero.

Degree of freedom 1 is u_r, degree of freedom 2 is u_z. Generalized axisymmetric elements with twist have an additional degree of freedom, 5, corresponding to the twist angle ϕ (in radians).

Abaqus/Standard does not automatically apply any boundary conditions to nodes located along the symmetry axis. You must apply radial or symmetry boundary conditions on these nodes if desired.

Point loads and moments should be given as the value integrated around the circumference; that is, the total value on the ring.

References:

- *Surface Elements*
- **SURFACE SECTION*
- **REBAR LAYER*

Element Types

Regular Axisymmetric Surface Elements

SFMAX1	2-node linear, without twist
SFMAX2	3-node quadratic, without twist

Active Degrees of Freedom

1, 2

Additional Solution Variables

None.

Generalized Axisymmetric Surface Elements

SFMGAX1	2-node linear, with twist

SFMGAX2 3-node quadratic, with twist

Active Degrees of Freedom

1, 2, 5

Additional Solution Variables

None.

Nodal Coordinates Required

R, *Z*

Element Property Definition

Input File Usage: Use the following option to define surface elements:

**SURFACE SECTION*

If rebar are being defined, use the following option in conjunction with the **SURFACE SECTION* option:

**REBAR LAYER*

Use the following option to define a mass density per unit area:

**SURFACE SECTION*, DENSITY=*number*

Abaqus/CAE Usage: Property module: **Create Section**: select **Shell** as the section **Category** and **Surface** as the section **Type**, **Rebar Layers** (optional)

You cannot define the mass per unit area for a surface section in Abaqus/CAE.

Element-Based Loading

Distributed Loads

Distributed loads are specified as described in *Distributed Loads*. Gravity and centrifugal loads apply only if the surface elements have rebar defined or if the elements have a defined density.

Load ID (*DLOAD): BR
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-2}
Description: Body force in the radial (1 or *r*) direction.

Load ID (*DLOAD): BZ
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-2}

Description: Body force in the axial (2 or z) direction.

Load ID (*DLOAD): BRNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-2}
Description: Nonuniform body force in the radial direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): BZNU
Abaqus/CAE Load/Interaction: Body force
Units: FL^{-2}
Description: Nonuniform body force in the axial direction with magnitude supplied via user subroutine *DLOAD*.

Load ID (*DLOAD): CENT
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-3} ($ML^{-2}T^{-2}$)

Description: Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit area, ω is the angular velocity). Since only axisymmetric deformation is allowed, the spin axis must be the z-axis.

Load ID (*DLOAD): CENTRIF
Abaqus/CAE Load/Interaction: Rotational body force
Units: T^{-2}

Description: Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity). Since only axisymmetric deformation is allowed, the spin axis must be the z-axis.

Load ID (*DLOAD): GRAV
Abaqus/CAE Load/Interaction: Gravity
Units: LT^{-2}
Description: Gravity loading in a specified direction (magnitude input as acceleration).

Load ID (*DLOAD): HP
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Hydrostatic pressure applied to the element reference surface and linear in global *Z*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure applied to the element reference surface. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): PNU
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform pressure applied to the element reference surface with magnitude supplied via user subroutine *DLOAD*. The pressure is positive in the direction of the positive element normal.

Load ID (*DLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: General traction on the element reference surface.

Load ID (*DLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-2}
Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Surface-Based Loading

Distributed Loads

Surface-based distributed loads are specified as described in *Distributed Loads*.

Load ID (*DSLOAD): HP
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Hydrostatic pressure applied to the element reference surface and linear in global *Z*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): P
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Pressure applied to the element reference surface. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): PNU
Abaqus/CAE Load/Interaction: Pressure
Units: FL^{-2}
Description: Nonuniform pressure applied to the element reference surface with magnitude supplied via user subroutine *DLOAD*. The pressure is positive in the direction opposite to the surface normal.

Load ID (*DSLOAD): ST
Abaqus/CAE Load/Interaction: Not supported
Units: FL^{-1}
Description: Surface tension on the element reference surface for regular elements.

Load ID (*DSLOAD): TRSHR
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}
Description: Shear traction on the element reference surface.

Load ID (*DSLOAD): TRSHRNU (S)
Abaqus/CAE Load/Interaction: Surface traction

Units: FL^{-2}

Description: Nonuniform shear traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

Load ID (*DSLOAD): TRVEC
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: General traction on the element reference surface.

Load ID (*DSLOAD): TRVECNU (S)
Abaqus/CAE Load/Interaction: Surface traction
Units: FL^{-2}

Description: Nonuniform general traction on the element reference surface with magnitude and direction supplied via user subroutine *UTRACLOAD*.

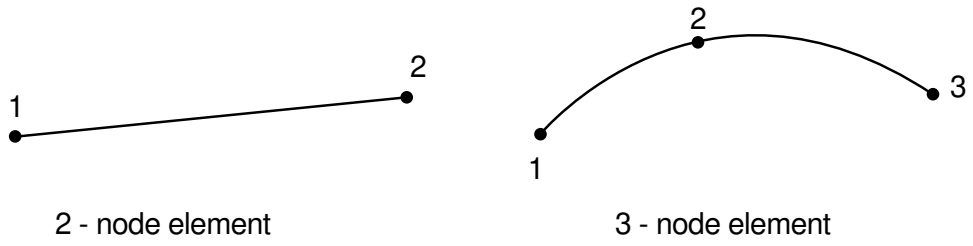
Incident Wave Loading

Surface-based incident wave loading is also available for these elements. See *Acoustic and Shock Loads*.

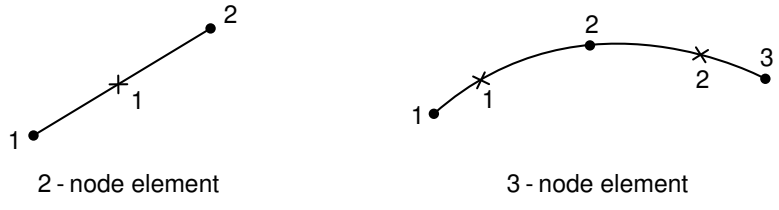
Element Output

Output is currently available only when the surface element is used to carry rebar layers. See *Defining Reinforcement* for details.

Node Ordering on Elements



Numbering of Integration Points for Output



Tube Support Elements

In this section:

- *Tube Support Elements*
- *Tube Support Element Library*

Tube Support Elements

Products: Abaqus/Standard

References:

- *Tube Support Element Library*
- **ITS*
- **DASHPOT*
- **FRICTION*
- **SPRING*

Overview

Tube support elements:

- are provided to model the interaction of a tube with a closely adjacent tube support, for cases where intermittent contact between the tube and the support may occur; and
- are made up of a spring/friction link (to simulate direct contact between the tube and the support) and a parallel dashpot (to simulate the effect of the fluid in the annulus between the tube and the support), as shown in *Figure 1*.

Details of the element formulations can be found in *Tube support elements*.

Typical Applications

An ITSCYL element can be used to model a drilled hole support (see *Figure 2*).

Several ITSUNI elements can be attached to the same node of the beam elements representing the tube to model the case of a tube support made up of a series of straight segments, as in an “egg-crate” design (see *Figure 3*).

Choosing an Appropriate Element

Two types of tube support elements are provided.

ITSUNI Elements

ITSUNI is a “unidirectional” element, which always acts in a fixed direction in space. One node of the element must be located on the axis of the tube, which is modeled using beam elements; and the other node must be located equidistant between the two parallel support plates. The support plates are built into the ITSUNI element definition.

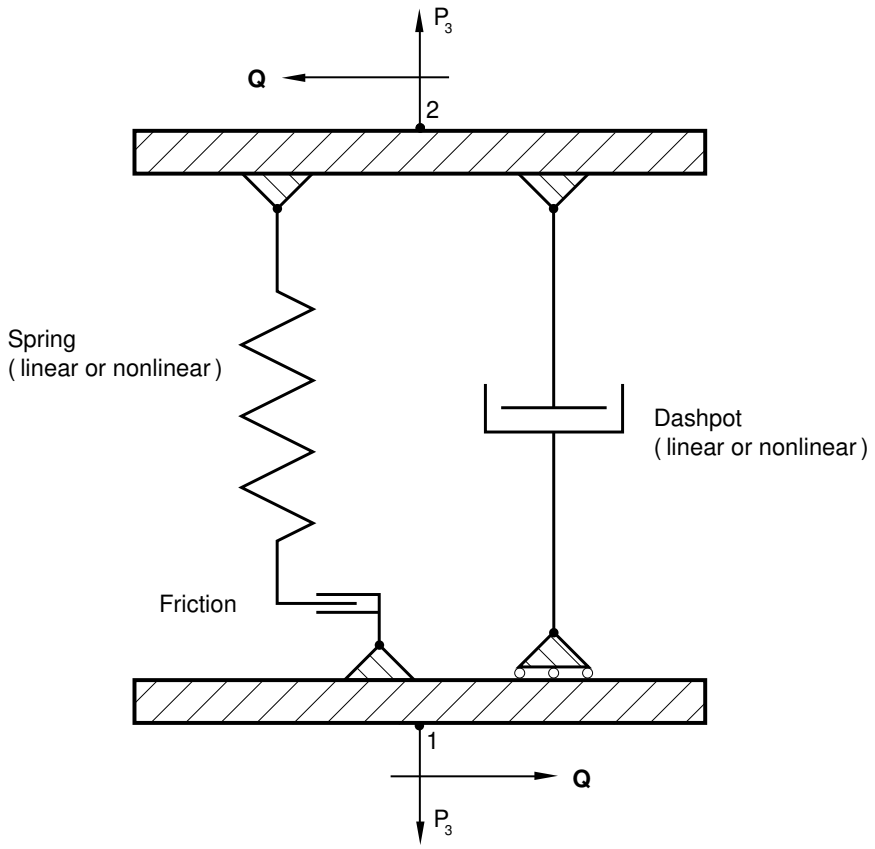


Figure 1: Tube support element behavior.

ITSCYL Elements

ITSCYL is a “cylindrical” element, which can be used to simulate the interaction between a circular tube and a circular hole. One node of the element must be located on the axis of the tube, which is modeled using beam elements, and the other node must be located at the center of the hole in the circular tube support plate. The circular hole is built into the ITSCYL element definition.

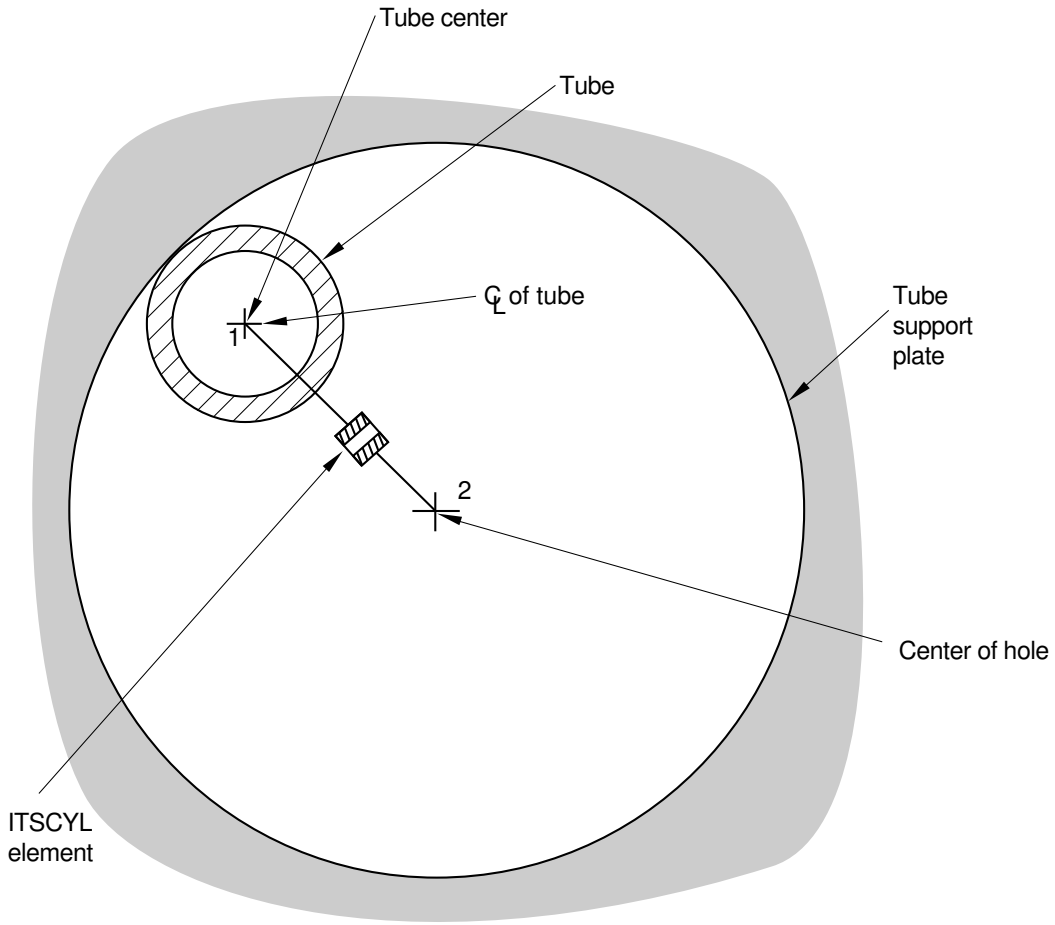


Figure 2: Use of an ITSCYL element for a drilled hole support.

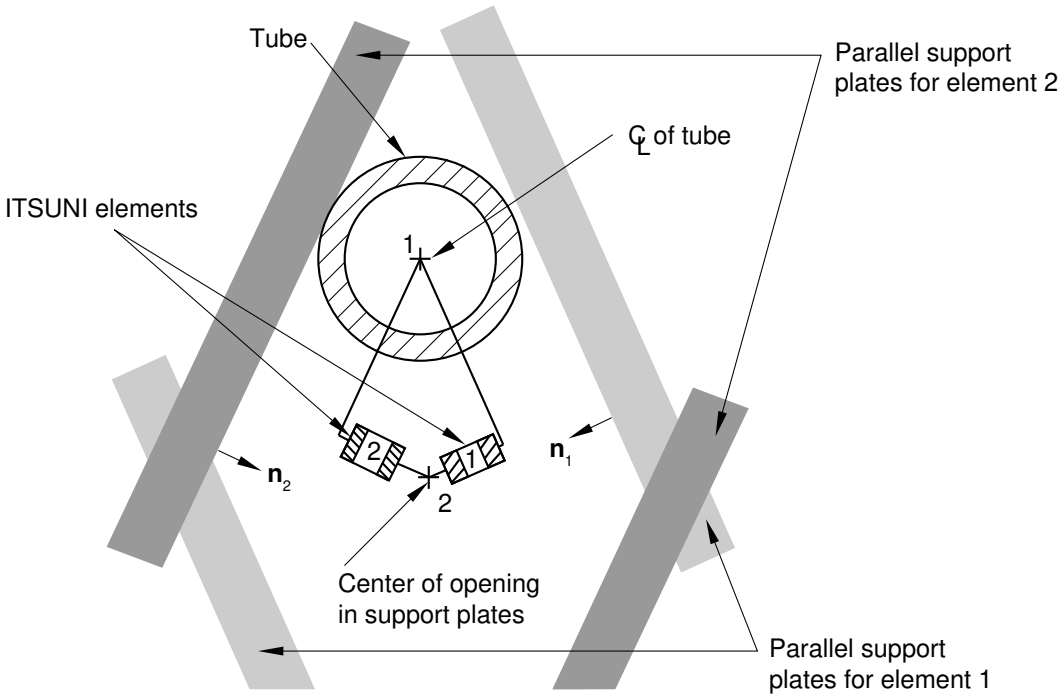


Figure 3: Use of ITSUNI elements for an “egg-crate” support.

Defining the Behavior of ITS Elements

You define the diameter of the tube and other geometric quantities that define the ITS element. You must associate these quantities with a set of ITS elements. In addition, you must define the behavior of the spring, friction link, and dashpot that make up a tube support element.

The spring behavior of an ITS element is shown in *Figure 4*. Relative displacements in the element are measured from the position where the tube and the hole in the support plate are aligned exactly—when the nodes of the element are at the same location. As indicated in *Figure 4*, the spring behavior of an ITS element is modified from that of the assigned spring definition to account for any clearance between the tube and support when the nodes of the element are at the same location. When there is no contact between the tube and the support, no force is transmitted by the spring; when the tube is in contact with the support, the force increases as the tube wall is deformed. This force can be modeled as a linear or a nonlinear function of the relative displacement between the axis of the tube and the center of the hole in the support.

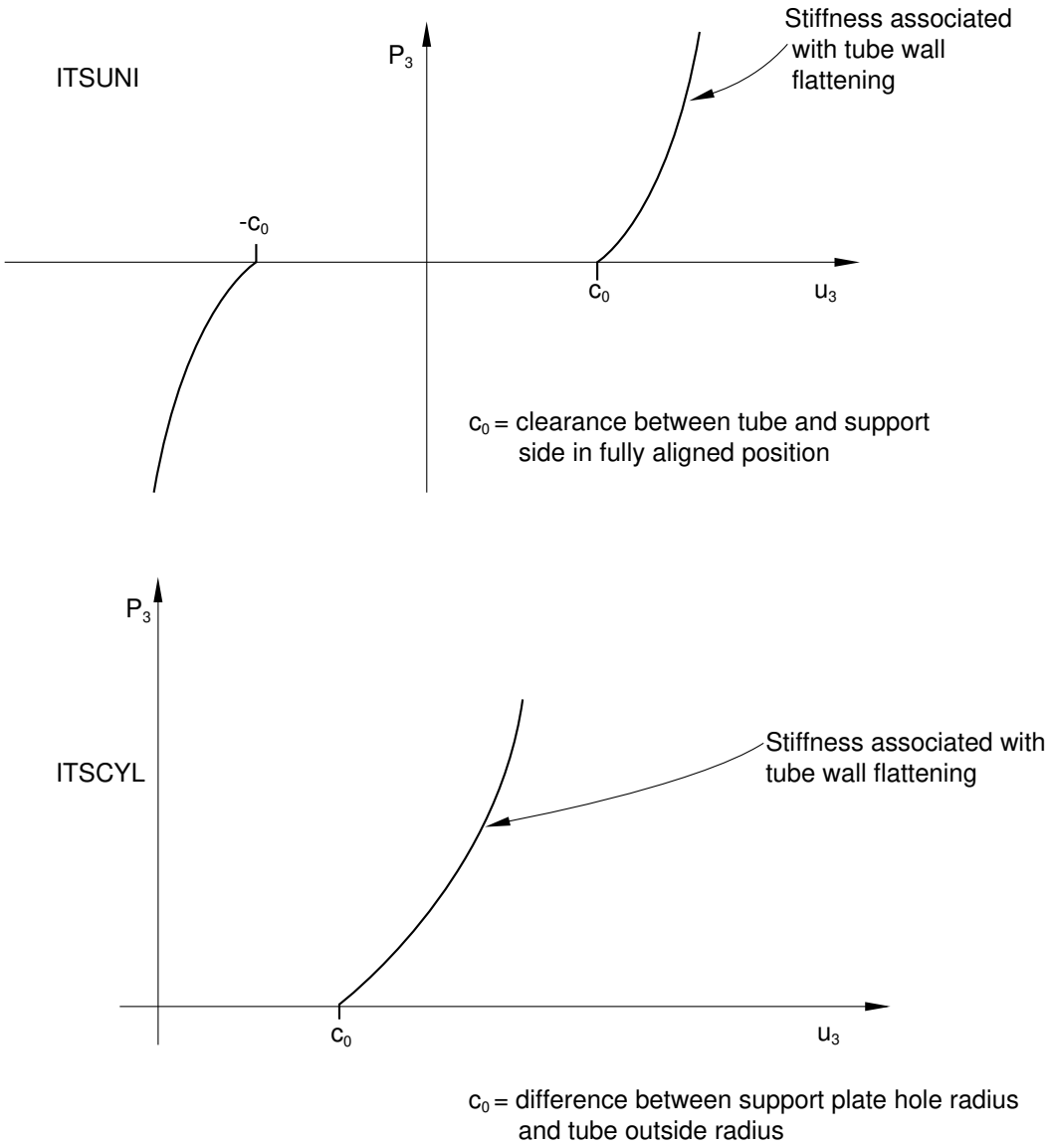


Figure 4: Nonlinear spring behavior in ITS elements to model clearance and tube flattening.

Friction between the tube and support will generate a moment at the tube node if the tube diameter is greater than zero and a moment at the hole node if the hole size is greater than zero. At least one of the following should be true for any node of an ITS element that will have a moment acting on it:

- the node should be associated with a beam or other element that can carry a moment;
- the nodal rotation should be set to zero with a boundary condition.

Input File Usage: Use the following options to define the behavior of ITS elements:

**ITS*, ELSET=*name*
**DASHPOT*
**SPRING*
**FRICTION*

Tube Support Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the tube support elements available in Abaqus/Standard.

References:

- *Tube Support Elements*
- **ITS*

Element Types

ITSUNI	Unidirectional tube support element
ITSCYL	Cylindrical geometry tube support element

Active Degrees of Freedom

1, 2, 3, 4, 5, 6

Additional Solution Variables

None.

Nodal Coordinates Required

X, *Y*, *Z*

Element Property Definition

Input File Usage: **ITS*

Element-Based Loading

None.

Element Output

S11	Total direct force in the element.

S12 Tangential (shear) force component, caused by friction, in the plane of the cross-section of the tube.

S13 Tangential (shear) force component, caused by friction, parallel to the axis of the tube.

The force in the spring link and the force in the dashpot are defined as generalized substresses and, therefore, are available as substress selections in the output options, as follows:

SS1 Force in the spring link.

SS2 Force in the dashpot.

The relative axial and tangential displacements corresponding to the forces above are chosen by requesting the corresponding "strains," except that "strain" component E13 is not defined in element type ITSCYL.

The relative tangential (shear) displacement components during slip are available as "plastic strain" components PE12 and PE13. The "equivalent plastic strain" is defined in these elements as

$$\Delta\overline{u}^{sl} = \sum_{\text{increments}} \sqrt{\left(\Delta u_1^{sl}\right)^2 + \left(\Delta u_2^{sl}\right)^2},$$

where Δu_1^{sl} and Δu_2^{sl} are the two relative tangential displacement components.

Nodes Associated with the Element

ITSUNI: Two nodes—one on the axis of the tube and one equidistant between the two parallel support plates.

ITSCYL: Two nodes—one on the axis of the tube and one at the center of the hole in the support plate.

User-Defined Elements

In this section:

- *User-Defined Elements*
- *User-Defined Element Library*

User-Defined Elements

Products: Abaqus/Standard Abaqus/Explicit

References:

- *User-Defined Element Library*
- *UEL*
- *UELMAT*
- *VUEL*
- *Accessing Abaqus Thermal Materials*
- *Accessing Abaqus Materials*
- **MATRIX*
- **UEL PROPERTY*
- **USER ELEMENT*

Overview

User-defined elements:

- can be finite elements in the usual sense of representing a geometric part of the model;
- can be feedback links, supplying forces at some points as functions of values of displacement, velocity, etc. at other points in the model;
- can be used to solve equations in terms of nonstandard degrees of freedom;
- can be linear or nonlinear; and
- can access selected materials from the Abaqus material library.

Assigning an Element Type Key to a User-Defined Element

You must assign an element type key to a user-defined element. The element type key must be of the form U*n* in Abaqus/Standard and VU*n* in Abaqus/Explicit, where *n* is a positive integer that identifies the element type uniquely. For example, you can define element types U1, U2, U3, VU1, VU7, etc. In Abaqus/Standard*n* must be less than 10000; while in Abaqus/Explicit*n* must be less than 9000.

The element type key is used to identify the element in the element definition. For general user elements the integer part of the identifier is provided in user subroutines *UEL*, *UELMAT* and *VUEL* so that you can distinguish between different element types.

Input File Usage: **USER ELEMENT*, TYPE=*element_type*

Invoking User-Defined Elements

User-defined elements are invoked in the same way as native Abaqus elements: you specify the element type, U*n* or VU*n*, and define element numbers and nodes associated with each element (see *Abaqus Model Definition*). User elements can be assigned to element sets in the usual way, for cross-reference to element property definitions, output requests, distributed load specifications, etc.

Material definitions (*Material Data Definition*) are relevant only to user-defined elements in Abaqus/Standard. If a material is assigned to a user-defined element (*Assigning an Abaqus Material to the User Element*), user subroutine *UELMAT* will be used to define the element response. User subroutine *UELMAT* allows access to selected Abaqus materials. If no material definition is specified, all material behavior must be defined in user

subroutines *UEL* and *VUEL*, based on user-defined material constants and on solution-dependent state variables associated with the element and calculated in the same subroutines. For linear user elements all material behavior must be defined through a user-defined stiffness matrix.

Input File Usage: Use the following options to invoke a user-defined element:

**USER ELEMENT*, TYPE=*element_type*
**ELEMENT*, TYPE=*element_type*

Defining the Active Degrees of Freedom at the Nodes

Any number of user element types can be defined and used in a model. Each user element can have any number of nodes, at each of which a specified set of degrees of freedom is used by the element. The activated degrees of freedom should follow the Abaqus convention (*Conventions*). In Abaqus/Standard this is important because the convergence criteria are based on the degrees of freedom numbers. In Abaqus/Explicit the activated degrees of freedom must follow the Abaqus convention because these are the only degrees of freedom that can be updated.

Abaqus always works in the global system when passing information to or from a user element. Therefore, the user element's stiffness, mass, etc. should always be defined with respect to global directions at its nodes, even if local transformations (*Transformed Coordinate Systems*) are applied to some of these nodes.

You define the ordering of the variables on a user element. The standard and recommended ordering is such that the degrees of freedom at the first node appear first, followed by the degrees of freedom at the second node, etc. For example, suppose that the user-defined element type is a planar beam with three nodes. The element uses degrees of freedom 1, 2, and 6 (u_x, u_y, and ϕ_z) at its first and last node and degrees of freedom 1 and 2 at its second (middle) node. In this case the ordering of variables on the element is:

Element variable number	Node	Degree of freedom
1	1	1
2	1	2
3	1	6
4	2	1
5	2	2
6	3	1
7	3	2
8	3	6

This ordering will be used in most cases. However, if you define an element matrix that does not have its degrees of freedom ordered in this way, you can change the ordering of the degrees of freedom as outlined below.

You specify the active degrees of freedom at each node of the element. If the degrees of freedom are the same at all of the element's nodes, you specify the list of degrees of freedom only once. Otherwise, you specify a new list of degrees of freedom each time the degrees of freedom at a node are different from those at previous nodes. Thus, different nodes of the element can use different degrees of freedom; this is especially useful when the element is being used in a coupled field problem so that, for example, some of its nodes have displacement degrees of freedom only, while others have displacement and temperature degrees of freedom. This method will produce an ordering of the element variables such that all of the degrees of freedom at the first node appear first, followed by the degrees of freedom at the second node, etc.

In Abaqus/Standard there are two ways to define element variable numbers that order the degrees of freedom on the element differently.

Since the user element can accept repeated node numbers when defining the nodal connectivity for the element, the element can be declared to have one node per degree of freedom on the element. For example, if the element

is a planar, 3-node triangle for stress analysis, it has three nodes, each of which has degrees of freedom 1 and 2. If all degrees of freedom 1 are to appear first in the element variables, the element can be defined with six nodes, the first three of which have degree of freedom 1, while nodes 4–6 have degree of freedom 2. The element variables would be ordered as follows:

Element variable number	Node	Degree of freedom
1	1	1
2	2	1
3	3	1
4	4	2
5	5	2
6	6	2

Alternatively, the user element variables can be defined so as to order the degrees of freedom on the element in any arbitrary fashion. You specify a list of degrees of freedom for the first node on the element. All nodes with a nodal connectivity number that is less than the next connectivity number for which a list of degrees of freedom is specified will have the first list of degrees of freedom. The second list of degrees of freedom will be used for all nodes until a new list is defined, etc. If a new list of degrees of freedom is encountered with a nodal connectivity number that is less than or equal to that given with the previous list, the previous list's degrees of freedom will be assigned through the last node of the element. This generation of degrees of freedom can be stopped before the last node on the element by specifying a nodal connectivity number with an empty (blank) list of degrees of freedom.

Example

The above procedure continues using this new list to define additional degrees of freedom according to the new node and degrees of freedom. For example, consider a 3-node beam that has degrees of freedom 1, 2, and 6 at nodes 1 and 3 and degrees of freedom 1 and 2 at node 2 (middle node). To order degrees of freedom 1 first, followed by 2, followed by 6, the following input could be used:

```
*USER ELEMENT
1
1, 2
1, 6
2,
3, 6
```

In this case the ordering of the variables on the element is:

Element variable number	Node	Degree of freedom
1	1	1
2	2	1
3	3	1
4	1	2
5	2	2
6	3	2
7	1	6
8	3	6

Requirements for Activated Degrees of Freedom in Abaqus/Explicit

There are the following additional requirements with respect to activated degrees of freedom on a user element of type VU*n*:

- Only degrees of freedom 1 through 6, 8, and 11 can be activated because these are the only degrees of freedom numbers that can be updated in Abaqus/Explicit. (In Abaqus/Standard degrees of freedom 1 through 30 can be used.)
- If one translational degree of freedom is activated at a node, all translational degrees of freedom up to the specified maximum number of coordinates must be activated at that node; moreover, the translational degrees of freedom at the node must be in consecutive order.
- In three-dimensional analyses, if one rotational degree of freedom is activated at a node, all three rotational degrees of freedom must be activated in consecutive order.

For example, if you define a 4-node three-dimensional user element that has translations and rotations active at the first and fourth nodes, temperature only at the second node, and translations and temperature at the third node, the following input could be used:

```
*USER ELEMENT
1,2,3,4,5,6
2,11
3,1,2,3,11
4,1,2,3,4,5,6
```

Rotation Update in Geometrically Nonlinear Analyses

If all three rotational degrees of freedom (4, 5, and 6) are used at a node in a geometrically nonlinear analysis, Abaqus assumes that these rotations are finite rotations. In this case the incremental values of these degrees of freedom are not simply added to the total values: the quaternion update formulae are used instead. Similarly, the corrections are not simply added to the incremental values. The update procedure is described in *Rotation variables* and is mentioned in *Conventions*.

To avoid the rotation update in a geometrically nonlinear analysis in Abaqus/Standard, you may define repeated node numbers in the nodal connectivity of the element such that at least one of the degrees of freedom 4, 5, or 6 is missing from the degree of freedom list at each node.

Visualizing User-Defined Elements in Abaqus/CAE

Plotting of user elements is not supported in Abaqus/CAE. However, if the user elements contain displacement degrees of freedom, they can be overlaid with standard elements; and model plots of these standard elements can be displayed, allowing you to see the shape of the user elements. If deformed mesh plots of the user elements are required, the material properties of the overlaying standard elements must be chosen so that the solution is not changed by including them. If this technique is used, nodes of the user element will be tied to nodes of the standard elements. Therefore, degrees of freedom 1, 2, and 3 in the user element must correspond to the displacement degrees of freedom at the nodes of the standard elements.

Defining a Linear User Element in Abaqus/Standard

Linear user elements can be defined only in Abaqus/Standard. In the simplest case a linear user element can be defined as a stiffness matrix and, if required, a mass matrix, as well as viscous and/or structural damping matrices. Stiffness and mass matrices can be read from a results file or defined directly. When damping is required, the linear user element must be defined directly. The damping matrices cannot be stored on a result file.

Reading the Element Matrices from an Abaqus/Standard Results File

To read the element matrices from an Abaqus/Standard results file, you must have written the stiffness and/or mass matrices in a previous analysis to the results file as element matrix output (*Element Matrix Output in*

Abaqus/Standard) or substructure matrix output (*Writing the Recovery Matrix, Reduced Stiffness Matrix, Mass Matrix, Load Case Vectors, and Gravity Vectors to a File*). Damping matrices cannot be written to the results file.

You must specify the element number, *n*, or substructure identifier, Z*n*, to which the matrices correspond. For models defined in terms of an assembly of part instances (*Assembly Definition*), the element numbers written to the results file are internal numbers generated by Abaqus/Standard (see *About Output*). A map between these internal numbers and the original element numbers and part instance names is provided in the data file of the analysis from which the element matrix output was written.

In addition, for element matrix output you must specify the step number and increment number at which the element matrix was written. These items are not required if a substructure whose matrix was output during its generation is used.

Input File Usage: **USER ELEMENT*, FILE=*name*, OLD ELEMENT=*n* or Z*n*, STEP=*n*, INCREMENT=*n*

Defining the Linear User Element by Specifying the Matrices Directly

If you define the stiffness, mass, or damping matrices directly, you must specify the number of nodes associated with the element.

Input File Usage: **USER ELEMENT*, LINEAR, NODES=*n*

Defining Whether or Not the Element Matrices Are Symmetric

If the element matrices are not symmetric, you can request that Abaqus/Standard use its nonsymmetric equation solution capability (see *Defining an Analysis*).

Input File Usage: **USER ELEMENT*, LINEAR, NODES=*n*, UNSYMM

Defining the Mass, Stiffness, or Damping Matrix

You define the element mass matrix, the element stiffness matrix, and the element viscous or structural damping matrix as needed. If the element is a heat transfer element, the “stiffness matrix” is the conductivity matrix and the “mass matrix” is the specific heat matrix.

You can define any applicable types of matrices simultanously.

You can read the mass and/or stiffness matrices from a file or define them directly. When damping matrices are needed, all of the matrices must be defined directly. In either case Abaqus/Standard reads four values per line, using F20 format. This format ensures that the data are read with adequate precision. Data written in E20.14 format can be read under this format.

Start with the first column of the matrix. Start a new line for each column. If you do not specify that the element matrix is unsymmetric, give the matrix entries from the top of each column to the diagonal term only: do not give the terms below the diagonal. If you specify that the element matrix is unsymmetric, give all terms in each column, starting from the top of the column.

Input File Usage: Use the following option to define the element mass matrix:

**MATRIX*, TYPE=MASS

Use the following option to define the element stiffness matrix:

*MATRIX, TYPE=STIFFNESS

Use the following option to define the element viscous damping matrix:

*MATRIX, TYPE=VISCOUS DAMPING

Use the following option to define the element structural damping matrix:

*MATRIX, TYPE=STRUCTURAL DAMPING

Use the following option to read the element mass or stiffness matrix from a file:

*MATRIX, TYPE=MASS or STIFFNESS, INPUT=*file_name*

For example, if the matrix is symmetric, the following data lines should be used:

A_{11}
A_{12}, A_{22}
A_{13}, A_{23}, A_{33}
$A_{14}, A_{24}, A_{34}, A_{44}$
$A_{15}, A_{25}, A_{35}, A_{45}$
A_{55}
$A_{16}, A_{26}, A_{36}, A_{46}$
A_{56}, A_{66}
Etc.

If the matrix is unsymmetric, the following data lines should be used:

$A_{11}, A_{21}, A_{31}, A_{41}$
$A_{51}, A_{61}, A_{71}, A_{81}$
…
…, A_{m1}
$A_{12}, A_{22}, A_{32}, A_{42}$
Etc.

where m is the size of the matrix and A_{ij} is the entry in the matrix for row i column j.

Geometrically Nonlinear Analysis

When a linear user element is used in a geometrically nonlinear analysis, the stiffness matrix provided will not be updated to account for any nonlinear effects such as finite rotations.

Defining the Element Properties

You must associate a property definition with every user element, even though no property values (except Rayleigh and/or structural damping factors) are associated with linear user elements.

Input File Usage: Use the following option to associate a property definition with a user element set:

*UEL PROPERTY, ELSET=*name*

Defining Damping for Dynamic Analyses

You can define the Rayleigh damping factors for dynamic analyses (*Implicit Dynamic Analysis Using Direct Integration*) only for linear user elements. These damping factors can be defined for mode-based transient analysis, mode-based steady-state analysis, and direct steady-state analysis for both the linear and nonlinear formulations. The Rayleigh damping factors are defined as

$$[\mathbf{C}] = \alpha [\mathbf{M}] + \beta [\mathbf{K}],$$

where $[\mathbf{C}]$ is the damping matrix, $[\mathbf{M}]$ is the mass matrix, $[\mathbf{K}]$ is the stiffness matrix, and α and β are the user-specified damping factors. See *Material Damping* for more information on Rayleigh damping.

You can define the structural damping factor (s) for direct and mode-based steady-state analyses. The damping matrix resulting from this factor is proportional to the user element stiffness matrix and is defined as

$$[\mathbf{S}] = s [\mathbf{K}].$$

Input File Usage: Use the following option to specify the Rayleigh damping factors:

**UEL PROPERTY*, ELSET=*name*, ALPHA=α, BETA=β

Use the following option to specify the structural damping factor:

**UEL PROPERTY*, ELSET=*name*, STRUCTURAL=s

Defining Loads

You can apply point loads, moments, fluxes, etc. to the nodes of linear user-defined elements in the usual way using concentrated loads and concentrated fluxes (*Concentrated Loads* and *Thermal Loads*).

Distributed loads and fluxes cannot be defined for linear user-defined elements.

Defining a General User Element

General user elements are defined in user subroutines *UEL* and *UELMAT* in Abaqus/Standard and in user subroutine *VUEL* in Abaqus/Explicit. *The implementation of user elements in user subroutines is recommended only for advanced users.*

Defining the Number of Nodes Associated with the Element

You must specify the number of nodes associated with a general user element. You can define "internal" nodes that are not connected to other elements.

Input File Usage: **USER ELEMENT*, NODES=*n*

Defining Whether or Not the Element Matrices Are Symmetric in Abaqus/Standard

If the contribution of the element to the Jacobian operator matrix of the overall Newton method is not symmetric (i.e., the element matrices are not symmetric), you can request that Abaqus/Standard use its nonsymmetric equation solution capability (see *Defining an Analysis*).

Input File Usage: **USER ELEMENT*, NODES=*n*, UNSYMM

Defining the Maximum Number of Coordinates Needed at Any Nodal Point

You can define the maximum number of coordinates needed in user subroutines *UEL*, *UELMAT*, or *VUEL* at any node point of the element. Abaqus assigns space to store this many coordinate values at all of the nodes associated with elements of this type. The default maximum number of coordinates at each node is 1.

Abaqus will change the maximum number of coordinates to be the maximum of the user-specified value or the value of the largest active degree of freedom of the user element that is less than or equal to 3. For example, if you specify a maximum number of coordinates of 1 and the active degrees of freedom of the user element are 2, 3, and 6, the maximum number of coordinates will be changed to 3. If you specify a maximum number of coordinates of 2 and the active degrees of freedom of the user element are 11 and 12, the maximum number of coordinates will remain as 2.

Input File Usage: **USER ELEMENT*, COORDINATES=*n*

Defining the Element Properties

You can define the number of properties associated with a particular user element and then specify their numerical values.

Specifying the Number of Property Values Required

Any number of properties can be defined to be used in forming a general user element. You can specify the number of integer property values required, *n*, and the number of real (floating point) property values required, *m*; the total number of values required is the sum of these two numbers. The default number of integer property values required is 0 and the default number of real property values required is 0.

Integer property values can be used inside user subroutines *UEL*, *UELMAT*, and *VUEL* as flags, indices, counters, etc. Examples of real (floating point) property values are the cross-sectional area of a beam or rod, thickness of a shell, and material properties to define the material behavior for the element.

Input File Usage: **USER ELEMENT*, I PROPERTIES=*n*, PROPERTIES=*m*

Specifying the Numerical Values of Element Properties

You must associate a user element property definition with each user-defined element, even if no property values are required. The property values specified in the property definition are passed into user subroutines *UEL*, *UELMAT*, and *VUEL* each time the subroutine is called for the user elements that are in the specified element set.

Input File Usage: Use the following option to associate a property definition with a user element set:

**UEL PROPERTY*, ELSET=*name*

To define the property values, enter all floating point values on the data lines first, followed immediately by the integer values. Eight values should be entered on all data lines except the last one, which may have fewer than eight values.

Assigning an Abaqus Material to the User Element

If the Abaqus material library is accessed from a user element, a material must be defined and assigned to the user element.

Input File Usage: Use the following option to associate a material with the user element:

**UEL PROPERTY*, MATERIAL=*name*

If this option is used, user subroutine *UELMAT* must be used to define the contribution of the element to the model. Otherwise, user subroutine *UEL* must be used.

Assigning an Orientation Definition

If the Abaqus material library is accessed from a user element, you can associate a material orientation definition (*Orientations*) with the user element. The orientation definition specifies a local coordinate system for material calculations in the element. The local coordinate system is assumed to be uniform in a given element and is based on the coordinates at the element centroid.

Input File Usage: Use the following option to associate an orientation definition with a user element:

**UEL PROPERTY*, ORIENTATION=*name*

Specifying the Element Type

If the Abaqus material library is accessed from a user element, the element type must be specified.

Input File Usage: Use the following option to define a three-dimensional element in a stress/ displacement or a heat transfer analysis:

**USER ELEMENT*, TENSOR=THREED

Use the following option to define a two-dimensional element in a heat transfer analysis:

**USER ELEMENT*, TENSOR=TWOD

Use the following option to define a plane strain element in a stress/ displacement analysis:

**USER ELEMENT*, TENSOR=PSTRAIN

Use the following option to define a plane stress element in a stress/ displacement analysis:

**USER ELEMENT*, TENSOR=PSTRESS

Specifying the Number of Integration Points

If the Abaqus material library is accessed from a user element, the number of integration points must be specified.

Input File Usage: Use the following option to specify the number of integration points:

**USER ELEMENT*, INTEGRATION=*n*

Defining the Number of Solution-Dependent Variables That Must Be Stored within the Element

You can define the number of solution-dependent state variables that must be stored within a general user element. The default number of variables is 1.

Examples of such variables are strains, stresses, section forces, and other state variables (for example, hardening measures in plasticity models) used in the calculations within the element. These variables allow quite general nonlinear kinematic and material behavior to be modeled. These solution-dependent state variables must be calculated and updated in user subroutines *UEL*, *UELMAT*, and *VUEL*.

As an example, suppose the element has four numerical integration points, at each of which you wish to store strain, stress, inelastic strain, and a scalar hardening variable to define the material state. Assume that the element is a three-dimensional solid, so that there are six components of stress and strain at each integration point. Then, the number of solution-dependent variables associated with each such element is $4 \times (6 \times 3 + 1) = 76$.

Input File Usage: **USER ELEMENT*, VARIABLES=*n*

Defining the Contribution of the Element to the Model in User Subroutine UEL

For a general user element in Abaqus/Standard, user subroutine *UEL* may be coded to define the contribution of the element to the model. Abaqus/Standard calls this routine each time any information about a user-defined element is needed. At each such call Abaqus/Standard provides the values of the nodal coordinates and of all solution-dependent nodal variables (displacements, incremental displacements, velocities, accelerations, etc.) at all degrees of freedom associated with the element, as well as values, at the beginning of the current increment, of the solution-dependent state variables associated with the element. Abaqus/Standard also provides the values of all user-defined properties associated with this element and a control flag array indicating what functions the user subroutine must perform. Depending on this set of control flags, the subroutine must define the contribution of the element to the residual vector, define the contribution of the element to the Jacobian (stiffness) matrix, update the solution-dependent state variables associated with the element, form the mass and damping matrix, and so on. Often, several of these functions must be performed in a single call to the routine. In mode-based analyses the user element is formulated only once and projected onto the eigensystem.

Formulation of an Element with User Subroutine UEL

The element's principal contribution to the model during general analysis steps is that it provides nodal forces F^N that depend on the values of the nodal variables u^M and on the solution-dependent state variables H^α within the element:

$$F^N = F^N\left(u^M, H^\alpha, \text{geometry, attributes, predefined field variables, distributed loads}\right).$$

Here we use the term “force” to mean that quantity in the variational statement that is conjugate to the basic nodal variable: physical force when the associated degree of freedom is physical displacement, moment when the associated degree of freedom is a rotation, heat flux when it is a temperature value, and so on. The signs of the forces in F^N are such that external forces provide positive nodal force values and “internal” forces caused by stresses, internal heat fluxes, etc. in the element provide negative nodal force values. For example, in the case of mechanical equilibrium of a finite element subject to surface tractions $\mathbf{t}$ and body forces $\mathbf{f}$ with stress $\boldsymbol{\sigma}$, and with interpolation $\delta\mathbf{u} = \mathbf{N}^N \delta u^N$, $\delta\boldsymbol{\varepsilon} = \boldsymbol{\beta}^N \delta u^N$,

$$F^N = \int_S \mathbf{N}^N \cdot \mathbf{t}\, dS + \int_V \mathbf{N}^N \cdot \mathbf{f}\, dV - \int_V \boldsymbol{\beta}^N : \boldsymbol{\sigma}\, dV.$$

In general procedures Abaqus/Standard solves the overall system of equations by Newton's method:

Solve
$$\widehat{K}^{NM} c^M = R^M,$$

Set
$$u^N = u^N + c^N,$$

Iterate

where R^N is the residual at degree of freedom *N* and

$$\widetilde{K}^{NM} = -\frac{dR^N}{du^M}$$

is the Jacobian matrix.

During such iterations you must define F^N, which is the element's contribution to the residual, R^N, and

$$-\frac{dF^N}{du^M},$$

which is the element's contribution to the Jacobian $\widetilde{K}^{NM}$. By writing the total derivative $-dF^N/du^M$, we imply that the element's contribution to $\widetilde{K}^{NM}$ should include all direct and indirect dependencies of the F^N on the u^M. For example, the H^α will generally depend on u^M; therefore, $-dF^N/du^M$ will include terms such as

$$-\frac{\partial F^N}{\partial H^\alpha}\frac{\partial H^\alpha}{\partial u^M}.$$

Use in Transient Analysis Procedures

In procedures such as transient heat transfer and dynamic analysis, the problem also involves time integration of rates of change of the nodal degrees of freedom. The time integration schemes used by Abaqus/Standard for the various procedures are described in more detail in the *Introduction: general*. For example, in transient heat transfer analysis, the backward difference method is used:

$$\dot{u}_{t+\Delta t} = \frac{1}{\Delta t}\left(u_{t+\Delta t} - u_t\right).$$

Therefore, if F^N depends on u^M and $\dot{u}^M$ (as would be the case if the user element includes thermal energy storage), the Jacobian contribution should include the term

$$-\frac{\partial F^N}{\partial \dot{u}^M}\left(\frac{d\dot{u}}{du}\right)_{t+\Delta t},$$

where $(d\dot{u}/du)_{t+\Delta t}$ is defined from the time integration procedure as $1/\Delta t$.

In all cases where Abaqus/Standard integrates first-order problems in time, the $\dot{u}^M$ are never stored because they are readily available as $\Delta u^M/\Delta t$, where $\Delta u^M = u^M_{t+\Delta t} - u^M_t$. However, for direct, implicit integration of dynamic systems (see *Implicit dynamic analysis*) Abaqus/Standard requires storage of $\dot{u}^M$ and $\ddot{u}^M$. These values are, therefore, passed into subroutine *UEL*. If the user element contains effects that depend on these time derivatives (damping and inertial effects), its Jacobian contribution will include

$$-\frac{\partial F^N}{\partial u^M} - \frac{\partial F^N}{\partial \dot{u}^M}\left(\frac{d\dot{u}}{du}\right)_{t+\Delta t} - \frac{\partial F^N}{\partial \ddot{u}^M}\left(\frac{d\ddot{u}}{du}\right)_{t+\Delta t}.$$

For the Hilber-Hughes-Taylor scheme

$$\left(\frac{d\dot{u}}{du}\right)_{t+\Delta t} = \frac{\gamma}{\beta\Delta t},$$
$$\left(\frac{d\ddot{u}}{du}\right)_{t+\Delta t} = \frac{1}{\beta\Delta t^2},$$

where β and γ are the (Newmark) parameters of the integration scheme. For backwark Euler time integration, the same expressions apply with β and γ equal to unity. The term $-\partial F^N/\partial \dot{u}^M$ is the element's damping matrix, and $-\partial F^N/\partial \ddot{u}^M$ is its mass matrix.

The Hilber-Hughes-Taylor scheme writes the overall dynamic equilibrium equations as

$$-M^{NM}\ddot{u}_{t+\Delta t} \quad + \quad (1+\alpha)\, G^N_{t+\Delta t} \quad - \quad \alpha\, G^N_t \quad = 0,$$

where G^N is the total force at degree of freedom N, excluding d'Alembert (inertia) forces. G^N is often referred to as the "static residual." Therefore, if a user element is to be used with Hilber-Hughes-Taylor time integration, the element's contribution F^N to the overall residual must be formulated in the same way. Since Abaqus/Standard provides information only at the time point at which *UEL* is called, this implies that each time *UEL* is called the H_α array must be used to recover G^N_t (and $G^N_{t^-}$ if half-increment residual calculations are required, where t^- indicates G^N from the beginning of the previous increment) and used to store $G_{t+\Delta t}$ (and G^N_t if half-increment residual calculations are required) for use in the next increment. This complication can be avoided if the numerical damping control parameter, α, for the dynamic step is set to zero; i.e., if the trapezoidal rule is used for integration of the dynamic equations (see *Implicit Dynamic Analysis Using Direct Integration* for details). This complication is also avoided with the backward Euler time integration operator because dynamic equilibrium is enforced at the end of the step.

If solution-dependent state variables (H^α) are used in the element, a suitable time integration method must be coded into subroutine *UEL* for these variables. Any of the u^N associated with the element that are not shared with standard Abaqus/Standard elements may be integrated in time by any suitable technique. If, in such cases, it is necessary to store values of u^N, $\dot{u}^N$, etc. at particular points in time, the solution-dependent state variable array, H_α, can be used for this purpose. Abaqus/Standard will still compute and store values of $\dot{u}^N$ and $\ddot{u}^N$ using the formulae associated with whatever time integrator it is using, but these values need not be used. To ensure accurate, stable time integration, you can control the size of the time increment used by Abaqus/Standard.

Constraints Defined with Lagrange Multipliers

Introduction of constraints with Lagrange multipliers should be avoided since Abaqus/Standard cannot detect such variables and avoid eigensolver problems by proper ordering of the equations.

Defining the Contribution of the Element to the Model in User Subroutine UELMAT

Alternatively, for a general user element in Abaqus/Standard, user subroutine *UELMAT* may be coded to define the contribution of the element to the model. User subroutine *UELMAT* is an enhanced version of user subroutine *UEL*; consequently, all the information provided for user subroutine *UEL* is also valid for user subroutine *UELMAT*. The enhancement allows you to access some of the material models from the Abaqus material library from *UELMAT*. *UELMAT* works only with a subset of procedures for which *UEL* is available:

- static;
- direct-integration dynamic;
- frequency extraction;
- steady-state uncouple heat transfer; and
- transient uncouple heat transfer.

User subroutine *UELMAT* will be called if an Abaqus material model is assigned to a user element (see *Assigning an Abaqus Material to the User Element* above); otherwise, user subroutine *UEL* will be called.

Accessing Abaqus Materials from User Subroutine UELMAT

Abaqus allows you to access some of the material models from the Abaqus material library from user subroutine *UELMAT*. The material models are accessed through the utility routines `MATERIAL_LIB_MECH` and `MATERIAL_LIB_HT` (*Accessing Abaqus Thermal Materials* and *Accessing Abaqus Materials*). Each time user subroutine *UELMAT* is called with the flags set to values that require computation of the right-hand-side vector and the element Jacobian, the material library must be called for each integration point, where the number of

integration points is specified in the element definition (*Specifying the Number of Integration Points*). The material models that are accessible from user subroutine *UELMAT* are:

- linear elastic model;
- hyperelastic model;
- Ramberg-Osgood model;
- classical metal plasticity models (Mises and Hill);
- extended Drucker-Prager model;
- modified Drucker-Prager/Cap plasticity model;
- porous metal plasticity model;
- elastomeric foam material model; and
- crushable foam plasticity model.

Defining the Contribution of the Element to the Model in User Subroutine VUEL

For a general user element in Abaqus/Explicit, user subroutine *VUEL* must be coded to define the contribution of the element to the model. Abaqus/Explicit calls this routine each time any information about a user-defined element is needed. At each such call Abaqus/Explicit provides the values of the nodal coordinates and of all solution-dependent nodal variables (displacements, velocities, accelerations, etc.) at all degrees of freedom associated with the element, as well as values of the solution-dependent state variables associated with the element at the beginning of the current increment. The incremental displacements are those obtained in a previous increment. Abaqus/Explicit also provides the values of all user-defined properties associated with this element and a control flag array indicating what functions the user subroutine must perform. Depending on this set of control flags, the subroutine must define the contribution of the element to the internal or external force/flux vector, form the mass/capacity matrix, update the solution-dependent state variables associated with the element, and so on.

The element's principal contribution to the model is that it provides nodal forces F^J that depend on the values of the nodal variables u^M, the rate of nodal variables $\dot{u}^M$, and on the solution-dependent state variables H^α within the element:

$$F^J = F^J\left(u^M, \dot{u}^M, H^\alpha, \text{geometry, attributes, predefined field variables, distributed loads}\right).$$

In addition, the element mass matrix M^{NJ} can be defined. Optionally, you can also define the external load contribution from the element due to specified distributed loading. In each increment Abaqus/Explicit solves for the accelerations at the end of the increment using

$$\ddot{u}^N_{(i)} = \left(M^{NJ}\right)^{-1}\left(P^J_{(i)} - F^J_{(i)}\right),$$

where P^J is the applied load vector. The solution (velocity, displacement) is then integrated in time using the central difference method

$$\dot{u}^N_{\left(i+\frac{1}{2}\right)} = \dot{u}^N_{\left(i-\frac{1}{2}\right)} + \frac{\Delta t_{(i+1)}+\Delta t_{(i)}}{2}\ddot{u}^N_{(i)},$$
$$u^N_{(i+1)} = u^N_{(i)} + \Delta t_{(i+1)}\dot{u}^N_{\left(i+\frac{1}{2}\right)}.$$

For coupled temperature/displacement elements the temperatures are computed at the beginning of the increment using

$$\dot{\theta}^N_{(i)} = \left(C^{NJ}\right)^{-1}\left(\mathbf{P}^J_{(i)} - F^J_{(i)}\right),$$

where C^{NJ} is the lumped capcitance matrix, P^J is the applied nodal source, and F^J is the internal flux vector. The temperature is integrated in time using the explicit forward-difference integration rule,

$$\theta^N{}_{(i+1)} = \theta^N_{(i)} + \Delta t_{(i+1)} \cdot \dot{\theta}^N_{(i)}.$$

More details can be found in *Explicit Dynamic Analysis* and *Fully Coupled Thermal-Stress Analysis*. The signs of the forces defined in F^J are such that external forces provide positive nodal force values and "internal" forces caused by stresses, damping effects, internal heat fluxes, etc. in the element provide negative nodal force values. Internal forces due to bulk viscosity are dependent on the scaled mass of the element. The necessary information (bulk viscosity constants and mass scaling factors) is passed into the user subroutine for this purpose.

Requirements for Defining the Mass Matrix

As explained in *Explicit Dynamic Analysis*, what makes the explicit time integration method efficient is that the mass inversion process is extremely effective. This is due to the fact that most of the nonzero entries in the mass matrix are located on the diagonal positions. The only exception is for rotational degrees of freedom in three-dimensional analyses in which case at each node an anisotropic rotary inertia (symmetric 3×3 tensor) can be defined. In these cases some of the nonzero entries in the mass matrix may be off-diagonal; but the inversion process is local and, hence, very effective. The mass matrix defined in user subroutine *VUEL* must adhere to these requirements as illustrated in detail in *VUEL*. If you specify a zero mass matrix or skip the definition of the mass matrix altogether, Abaqus/Explicit issues an error message.

The definition of a realistic mass matrix is not mandatory, but it is strongly recommended. If you choose to not define a realistic mass matrix using the user subroutine, you must provide realistic mass, rotary inertia, heat capacity, etc. at all nodes and at all degrees of freedom associated with the user element. This can be accomplished by various means, such as by defining mass and rotary inertia elements at the nodes or by connecting the user element to other elements for which density, heat capacity, etc. was specified.

Mass is computed only once at the beginning of the analysis. Consequently, the mass of the user element cannot be changed arbitrarily during the analysis. If necessary, mass scaling is applied accordingly to ensure the requested time incrementation.

Definition of the Stable Time Increment

Since the central difference operator is conditionally stable, the time increments in Abaqus/Explicit must be somewhat smaller than the stable time increment. You must provide an accurate estimate for the stable time increment associated with the user element. This scalar value is highly dependent on the element formulation, and sophisticated coding may be required inside the user subroutine to obtain a reliable estimate. A conservative estimate will reduce the time increment size for the entire analysis and, hence, lead to longer analysis times.

Defining Loads

You can apply point loads, moments, fluxes, etc. to the nodes of general user-defined elements in the usual way, using concentrated loads and concentrated fluxes (*Concentrated Loads* and *Thermal Loads*).

You can also define distributed loads and fluxes for general user-defined elements (*Distributed Loads* and *Thermal Loads*). These loads require a load type key. For user-defined elements, you can define load type keys of the forms U*n* and, in Abaqus/Standard, U*n*NU, where *n* is any positive integer.

If the load type key is of the form U*n*, the load magnitude is defined directly and follows the standard Abaqus conventions with respect to its amplitude variation as a function of time. In Abaqus/Standard, if the load key is of the form U*n*NU, all of the load definition will be accomplished inside subroutine *UEL* and *UELMAT*. Each time Abaqus/Standard calls subroutine *UEL* or *UELMAT*, it tells the subroutine how many distributed loads/fluxes are currently active. For each active load or flux of type U*n*Abaqus/Standard gives the current magnitude and current increment in magnitude of the load. The coding in subroutine *UEL* or *UELMAT* must distribute the loads

into consistent equivalent nodal forces and, if necessary, provide their contribution to the Jacobian matrix—the "load stiffness matrix."

In Abaqus/Explicit only load keys of the form U*n* can be used, and they can be used only for distributed loads (however, thermal fluxes can be defined in the coding in subroutine *VUEL*). Each time Abaqus/Explicit calls subroutine *VUEL*, it tells the subroutine which load number is currently active and the current magnitude of the load. The coding in subroutine *VUEL* must distribute the loads into consistent equivalent nodal forces.

Defining Output

All quantities to be output must be saved as solution-dependent state variables. In Abaqus/Standard the solution-dependent state variables can be printed or written to the results file using output variable identifier SDV (*Abaqus/Standard Output Variable Identifiers*).

The components of solution-dependent state variables that belong to a user element are not available in Abaqus/CAE. You can write output to separate files in a table format that can be accessed in Abaqus/CAE to produce history output.

Currently element output to the output database is not supported for user-defined elements.

Defining Wave Kinematic Data

A utility routine `GETWAVE` is provided in user subroutine *UEL* to access the wave kinematic data defined for an Abaqus/Aqua analysis (*Abaqus/Aqua Analysis*). This utility is discussed in *Obtaining Wave Kinematic Data in an Abaqus/Aqua Analysis*, where the arguments to `GETWAVE` and the syntax for its use are defined.

Use in Contact

Only node-based surfaces (*Node-Based Surface Definition*) can be created on user-defined elements. Hence, these elements can be used to define only secondary surfaces in a contact analysis. In Abaqus/Explicit the user elements will not be included in the general contact algorithm automatically. Node-based surfaces can be defined using these nodes and then included in the general contact definition.

Import of User Elements

User elements cannot be imported from an Abaqus/Standard analysis into an Abaqus/Explicit analysis or vice versa. Equivalent user elements can be defined in both products to overcome this limitation. However, the state variables associated with these elements will not be communicated.

User-Defined Element Library

Products: Abaqus/Standard Abaqus/Explicit

Overview

This section provides a reference to the user-defined elements available in Abaqus/Standard and Abaqus/Explicit.

References:

- *User-Defined Elements*
- *UEL*
- *UELMAT*
- *VUEL*
- **MATRIX*
- **UEL PROPERTY*
- **USER ELEMENT*

Element Types

U*n* *n* must be a positive integer ($0 < n < 10000$) that will define the element type uniquely in Abaqus/Standard

VU*n* *n* must be a positive integer ($0 < n < 10000$) that will define the element type uniquely in Abaqus/Explicit

Active Degrees of Freedom

As defined in the user element definition.

Additional Solution Variables

You can define solution variables associated with nodes that are not connected to other elements. However, in Abaqus/Standard, definition of constraints with Lagrange multipliers in user elements should be avoided because of potential equation solver problems.

In Abaqus/Explicit definition of constraints with Lagrange multipliers is not possible because the stable time increment would decrease to infinitesimally small values.

Nodal Coordinates Required

None required for linear user elements.

As needed in user subroutines *UEL*, *UELMAT*, or *VUEL* for general user elements. The maximum number of coordinates per node is specified in the user element definition (see *Defining the Maximum Number of Coordinates Needed at Any Nodal Point*). The first coordinate entries at each node should correspond to the standard Abaqus convention (*X*, *Y*, *Z* or *r*, *z* for axisymmetric elements).

Element Property Definition

For a linear user element the properties are the stiffness and mass, defined via user-defined matrices or read from an Abaqus/Standard results file. If necessary, you can specify Rayleigh damping values for linear user elements in the element property definition.

For a general user element defined via user subroutines *UEL*, *UELMAT*, or *VUEL*, you define the number of element properties in the user element definition and provide the numerical values in the element property definition. The definition of these properties depends on your coding in subroutine *UEL*, *UELMAT*, or *VUEL*.

Input File Usage: **UEL PROPERTY*

Element-Based Loading

None for linear user elements.

U*n*: Distributed load or flux whose magnitude is given via distributed load or distributed flux loading definitions (see *Distributed Loads* or *Thermal Loads*) for a general user element. *n* must be a positive integer that is passed into user subroutines *UEL*, *UELMAT*, or *VUEL* to identify the particular load type.

U*n*NU: Available in Abaqus/Standard only. Distributed load or flux that is completely defined as equivalent nodal values inside user subroutine *UEL* or *UELMAT* for a general user element. *n* must be a positive integer: $-n$ will be passed into subroutine *UEL* or *UELMAT* when such a load is active to identify the load type. The minus sign on *n* indicates that the load is of type NU.

Element Output

For a linear user element there are no stress or strain components since the element only appears as a stiffness and mass.

For a general user element any stress, strain, or other solution-dependent variables within the element must be defined as solution-dependent state variables by your coding within subroutine *UEL*, *UELMAT*, or *VUEL*. In Abaqus/Standard they can be printed or written to the results file using output variable SDV.

Currently element output to the output database is not supported for user-defined elements.

Node Ordering on Elements

As defined in the user element definition.

Element Indexes

These indexes list the elements that apply to a particular Abaqus product.

In this section:

- *Abaqus/Standard Element Index*
- *Abaqus/Explicit Element Index*

Abaqus/Standard Element Index

This index provides a reference to all of the element types that are available in Abaqus/Standard.

Elements are listed in alphabetical order, where numerical characters precede the letter “A” and two-digit numbers are put in numerical, rather than “alphabetical,” order. Thus, AC1D2 precedes ACAX4, and AC3D20 follows AC3D8.

For certain options, such as contact and surface-based distributing coupling, Abaqus may generate internal elements (such as IDCOUP3D for surface-based distributing coupling). These internal element names are not included in the index below but may appear in an output database (`.odb`) or data (`.dat`) file.

AC1D2 2-node acoustic link

AC1D3 3-node acoustic link

AC2D3 3-node linear 2D acoustic triangle

AC2D4 4-node linear 2D acoustic quadrilateral

AC2D6 6-node quadratic 2D acoustic triangular prism

AC2D8 8-node quadratic 2D acoustic quadrilateral

AC3D4 4-node linear acoustic tetrahedron

AC3D5 5-node linear acoustic pyramid

AC3D6 6-node linear acoustic triangular prism

AC3D8 8-node linear acoustic brick

AC3D10 10-node quadratic acoustic tetrahedron

AC3D15 15-node quadratic acoustic triangular prism

AC3D20 20-node quadratic acoustic brick

ACAX3 3-node linear axisymmetric acoustic triangle

ACAX4 4-node linear axisymmetric acoustic quadrilateral

ACAX6 6-node quadratic axisymmetric acoustic triangle

ACAX8 8-node quadratic axisymmetric acoustic quadrilateral

ACIN2D2 2-node linear 2D acoustic infinite element

ACIN2D3 3-node quadratic 2D acoustic infinite element

ACIN3D3	3-node linear 3D acoustic infinite element
ACIN3D4	4-node linear 3D acoustic infinite element
ACIN3D6	6-node quadratic 3D acoustic infinite element
ACIN3D8	8-node quadratic 3D acoustic infinite element
ACINAX2	2-node linear axisymmetric acoustic infinite element
ACINAX3	3-node quadratic axisymmetric acoustic infinite element
ASI1	1-node acoustic interface element
ASI2	2-node linear 2D acoustic interface element (this element has been renamed to ASI2D2)
ASI2A	2-node linear axisymmetric acoustic interface element (this element has been renamed to ASIAX2)
ASI2D2	2-node linear 2D acoustic interface element
ASI2D3	3-node quadratic 2D acoustic interface element
ASI3	3-node quadratic 2D acoustic interface element (this element has been renamed to ASI2D3)
ASI3A	3-node quadratic axisymmetric acoustic interface element (this element has been renamed to ASIAX3)
ASI3D3	3-node linear 3D acoustic interface element
ASI3D4	4-node linear 3D acoustic interface element
ASI3D6	6-node quadratic 3D acoustic interface element
ASI3D8	8-node quadratic 3D acoustic interface element
ASI4	4-node linear 3D acoustic interface element (this element has been renamed to ASI3D4)
ASI8	8-node quadratic 3D acoustic interface element (this element has been renamed to ASI3D8)
ASIAX2	2-node linear axisymmetric acoustic interface element
ASIAX3	3-node quadratic axisymmetric acoustic interface element

B21	2-node linear beam in a plane
B21H	2-node linear beam in a plane, hybrid formulation
B22	3-node quadratic beam in a plane
B22H	3-node quadratic beam in a plane, hybrid formulation
B23	2-node cubic beam in a plane
B23H	2-node cubic beam in a plane, hybrid formulation
B31	2-node linear beam in space
B31H	2-node linear beam in space, hybrid formulation
B31OS	2-node linear open-section beam in space
B31OSH	2-node linear open-section beam in space, hybrid formulation
B32	3-node quadratic beam in space
B32H	3-node quadratic beam in space, hybrid formulation
B32OS	3-node quadratic open-section beam in space
B32OSH	3-node quadratic open-section beam in space, hybrid formulation
B33	2-node cubic beam in space
B33H	2-node cubic beam in space, hybrid formulation
C3D4	4-node linear tetrahedron
C3D4A	4-node linear poroelastic acoustic tetrahedron
C3D4E	4-node linear piezoelectric tetrahedron
C3D4H	4-node linear tetrahedron, hybrid, linear pressure
C3D4P	4-node linear coupled pore pressure element
C3D4PH	4-node linear coupled pore pressure element, hybrid with constant pressure
C3D4PHT	4-node linear coupled pore pressure and temperature element, hybrid with constant pressure

C3D4PT	4-node linear coupled pore pressure and temperature element
C3D4T	4-node thermally coupled tetrahedron, linear displacement and temperature
C3D5	5-node linear pyramid
C3D5H	5-node linear pyramid, constant pressure
C3D6	6-node linear triangular prism
C3D6A	6-node linear poroelastic acoustic triangular prism
C3D6E	6-node linear piezoelectric triangular prism
C3D6H	6-node linear triangular prism, hybrid, constant pressure
C3D6HT	6-node linear coupled temperature element, hybrid with constant pressure
C3D6P	6-node linear coupled pore pressure element
C3D6PH	6-node linear coupled pore pressure element, hybrid with constant pressure
C3D6PHT	6-node linear coupled pore pressure and temperature element, hybrid with constant pressure
C3D6PT	6-node linear coupled pore pressure and temperature element
C3D6T	6-node thermally coupled triangular prism, linear displacement and temperature
C3D8	8-node linear brick
C3D8A	8-node linear poroelastic acoustic brick
C3D8E	8-node linear piezoelectric brick
C3D8H	8-node linear brick, hybrid, constant pressure
C3D8HT	8-node thermally coupled brick, trilinear displacement and temperature, hybrid, constant pressure
C3D8I	8-node linear brick, incompatible modes
C3D8IT	8-node linear brick, displacement and temperature, incompatible modes
C3D8IH	8-node linear brick, hybrid, linear pressure, incompatible modes

C3D8IHT	8-node linear brick, displacement and temperature, hybrid with linear pressure, incompatible modes
C3D8P	8-node brick, trilinear displacement, trilinear pore pressure
C3D8PH	8-node brick, trilinear displacement, trilinear pore pressure, hybrid, constant pressure
C3D8PHT	8-node brick, trilinear displacement, trilinear pore pressure, trilinear temperature, hybrid, constant pressure
C3D8PT	8-node brick, trilinear displacement, trilinear pore pressure, trilinear temperature
C3D8R	8-node linear brick, reduced integration, hourglass control
C3D8RH	8-node linear brick, hybrid, constant pressure, reduced integration, hourglass control
C3D8RHT	8-node thermally coupled brick, trilinear displacement and temperature, reduced integration, hourglass control, hybrid, constant pressure
C3D8RP	8-node brick, trilinear displacement, trilinear pore pressure, reduced integration
C3D8RPH	8-node brick, trilinear displacement, trilinear pore pressure, reduced integration, hybrid, constant pressure
C3D8RPHT	8-node brick, trilinear displacement, trilinear pore pressure, trilinear temperature, reduced integration, hybrid, constant pressure
C3D8RPT	8-node brick, trilinear displacement, trilinear pore pressure, trilinear temperature, reduced integration
C3D8RT	8-node thermally coupled brick, trilinear displacement and temperature, reduced integration, hourglass control
C3D8S	8-node linear brick, improved surface stress visualization
C3D8HS	8-node linear brick, hybrid, constant pressure, improved surface stress visualization
C3D8T	8-node thermally coupled brick, trilinear displacement and temperature
C3D10	10-node quadratic tetrahedron
C3D10E	10-node quadratic piezoelectric tetrahedron
C3D10H	10-node quadratic tetrahedron, hybrid, constant pressure
C3D10HT	10-node triquadratic displacement, trilinear temperature, hybrid with constant pressure
C3D10HS	10-node general-purpose quadratic tetrahedron, improved surface stress visualization

C3D10M	10-node modified tetrahedron, hourglass control
C3D10MH	10-node modified quadratic tetrahedron, hybrid, linear pressure, hourglass control
C3D10MHT	10-node thermally coupled modified quadratic tetrahedron, hybrid, linear pressure, hourglass control
C3D10MP	10-node modified displacement and pore pressure tetrahedron, hourglass control
C3D10MPH	10-node modified displacement and pore pressure tetrahedron, hybrid, linear pressure, hourglass control
C3D10MPT	10-node modified displacement, pore pressure, and temperature tetrahedron, linear pressure, hourglass control
C3D10MT	10-node modified thermally coupled tetrahedron, hourglass control
C3D10P	10-node triquadratic displacement, trilinear pore pressure
C3D10PH	10-node triquadratic displacement, trilinear pore pressure, hybrid with constant pressure
C3D10PHT	10-node triquadratic displacement, trilinear pore pressure, trilinear temperature, hybrid with constant pressure
C3D10PT	10-node triquadratic displacement, trilinear pore pressure, trilinear temperature
C3D10T	10-node triquadratic displacement, trilinear temperature
C3D15	15-node quadratic triangular prism
C3D15E	15-node quadratic piezoelectric triangular prism
C3D15H	15-node quadratic triangular prism, hybrid, linear pressure
C3D15V	15 to 18-node triangular prism
C3D15VH	15 to 18-node triangular prism, hybrid, linear pressure
C3D20	20-node quadratic brick
C3D20E	20-node quadratic piezoelectric brick
C3D20H	20-node quadratic brick, hybrid, linear pressure
C3D20HT	20-node thermally coupled brick, triquadratic displacement, trilinear temperature, hybrid, linear pressure

C3D20P	20-node brick, triquadratic displacement, trilinear pore pressure
C3D20PH	20-node brick, triquadratic displacement, trilinear pore pressure, hybrid, linear pressure
C3D20R	20-node quadratic brick, reduced integration
C3D20RE	20-node quadratic piezoelectric brick, reduced integration
C3D20RH	20-node quadratic brick, hybrid, linear pressure, reduced integration
C3D20RHT	20-node thermally coupled brick, triquadratic displacement, trilinear temperature, hybrid, linear pressure, reduced integration
C3D20RP	20-node brick, triquadratic displacement, trilinear pore pressure, reduced integration
C3D20RPH	20-node brick, triquadratic displacement, trilinear pore pressure, hybrid, linear pressure, reduced integration
C3D20RT	20-node thermally coupled brick, triquadratic displacement, trilinear temperature, reduced integration
C3D20T	20-node thermally coupled brick, triquadratic displacement, trilinear temperature
C3D27	21 to 27-node brick
C3D27H	21 to 27-node brick, hybrid, linear pressure
C3D27R	21 to 27-node brick, reduced integration
C3D27RH	21 to 27-node brick, hybrid, linear pressure, reduced integration
CAX3	3-node linear axisymmetric triangle
CAX3E	3-node linear axisymmetric piezoelectric triangle
CAX3H	3-node linear axisymmetric triangle, hybrid, constant pressure
CAX3T	3-node axisymmetric thermally coupled triangle, linear displacement and temperature
CAX4	4-node bilinear axisymmetric quadrilateral
CAX4E	4-node bilinear axisymmetric piezoelectric quadrilateral
CAX4H	4-node bilinear axisymmetric quadrilateral, hybrid, constant pressure
CAX4HT	4-node axisymmetric thermally coupled quadrilateral, bilinear displacement and temperature, hybrid, constant pressure

CAX4I	4-node bilinear axisymmetric quadrilateral, incompatible modes
CAX4IH	4-node bilinear axisymmetric quadrilateral, hybrid, linear pressure, incompatible modes
CAX4P	4-node axisymmetric quadrilateral, bilinear displacement, bilinear pore pressure
CAX4PH	4-node axisymmetric quadrilateral, bilinear displacement, bilinear pore pressure, hybrid, constant pressure
CAX4PT	4-node axisymmetric quadrilateral, bilinear displacement, bilinear pore pressure, bilinear temperature
CAX4R	4-node bilinear axisymmetric quadrilateral, reduced integration, hourglass control
CAX4RH	4-node bilinear axisymmetric quadrilateral, hybrid, constant pressure, reduced integration, hourglass control
CAX4RHT	4-node thermally coupled axisymmetric quadrilateral, bilinear displacement and temperature, reduced integration, hourglass control
CAX4RP	4-node axisymmetric quadrilateral, bilinear displacement, bilinear pore pressure, reduced integration
CAX4RPH	4-node axisymmetric quadrilateral, bilinear displacement, bilinear pore pressure, hybrid, constant pressure, reduced integration
CAX4RPHT	4-node axisymmetric quadrilateral, bilinear displacement, bilinear pore pressure, bilinear temperature, hybrid, constant pressure, reduced integration
CAX4RPT	4-node axisymmetric quadrilateral, bilinear displacement, bilinear pore pressure, bilinear temperature, reduced integration
CAX4RT	4-node thermally coupled axisymmetric quadrilateral, bilinear displacement and temperature, hybrid, constant pressure, reduced integration, hourglass control
CAX4T	4-node axisymmetric thermally coupled quadrilateral, bilinear displacement and temperature
CAX6	6-node quadratic axisymmetric triangle
CAX6E	6-node quadratic axisymmetric piezoelectric triangle
CAX6H	6-node quadratic axisymmetric triangle, hybrid, linear pressure
CAX6M	6-node modified axisymmetric triangle, hourglass control

CAX6MH 6-node modified quadratic axisymmetric triangle, hybrid, linear pressure, hourglass control

CAX6MHT 6-node modified axisymmetric thermally coupled triangle, hybrid, linear pressure, hourglass control

CAX6MP 6-node modified displacement and pore pressure axisymmetric triangle, hourglass control

CAX6MPH 6-node modified displacement and pore pressure axisymmetric triangle, hybrid, linear pressure, hourglass control

CAX6MT 6-node modified axisymmetric thermally coupled triangle, hourglass control

CAX8 8-node biquadratic axisymmetric quadrilateral

CAX8E 8-node biquadratic axisymmetric piezoelectric quadrilateral

CAX8H 8-node biquadratic axisymmetric quadrilateral, hybrid, linear pressure

CAX8HT 8-node axisymmetric thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, hybrid, linear pressure

CAX8P 8-node axisymmetric quadrilateral, biquadratic displacement, bilinear pore pressure

CAX8PH 8-node axisymmetric quadrilateral, biquadratic displacement, bilinear pore pressure, hybrid, linear pressure

CAX8R 8-node biquadratic axisymmetric quadrilateral, reduced integration

CAX8RE 8-node biquadratic axisymmetric piezoelectric quadrilateral, reduced integration

CAX8RH 8-node biquadratic axisymmetric quadrilateral, hybrid, linear pressure, reduced integration

CAX8RHT 8-node axisymmetric thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, hybrid, linear pressure, reduced integration

CAX8RP 8-node axisymmetric quadrilateral, biquadratic displacement, bilinear pore pressure, reduced integration

CAX8RPH 8-node axisymmetric quadrilateral, biquadratic displacement, bilinear pore pressure, hybrid, linear pressure, reduced integration

CAX8RT 8-node axisymmetric thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, reduced integration

CAX8T	8-node axisymmetric thermally coupled quadrilateral, biquadratic displacement, bilinear temperature
CAXA4N	Bilinear asymmetric-axisymmetric, Fourier quadrilateral with 4 nodes per r–z plane
CAXA4HN	Bilinear asymmetric-axisymmetric, Fourier quadrilateral with 4 nodes per r–z plane, constant Fourier pressure, hybrid
CAXA4RN	Bilinear asymmetric-axisymmetric, Fourier quadrilateral with 4 nodes per r–z plane, reduced integration in r–z planes, hourglass control
CAXA4RHN	Bilinear asymmetric-axisymmetric, Fourier quadrilateral with 4 nodes per r–z plane, constant Fourier pressure, hybrid, reduced integration in r–z planes
CAXA8N	Biquadratic asymmetric-axisymmetric, Fourier quadrilateral with 8 nodes per r–z plane
CAXA8HN	Biquadratic asymmetric-axisymmetric, Fourier quadrilateral with 8 nodes per r–z plane, linear Fourier pressure, hybrid
CAXA8PN	Biquadratic asymmetric-axisymmetric, Fourier quadrilateral with 8 nodes per r–z plane, bilinear Fourier pore pressure
CAXA8RN	Biquadratic asymmetric-axisymmetric, Fourier quadrilateral with 8 nodes per r–z plane, reduced integration in r–z planes
CAXA8RHN	Biquadratic asymmetric-axisymmetric, Fourier quadrilateral with 8 nodes per r–z plane, linear Fourier pressure, hybrid, reduced integration in r–z planes
CAXA8RPN	Biquadratic asymmetric-axisymmetric, Fourier quadrilateral with 8 nodes per r–z plane, bilinear Fourier pore pressure, reduced integration in r–z planes
CCL9	9-node cylindrical prism
CCL9H	9-node cylindrical hybrid prism
CCL12	12-node cylindrical brick
CCL12H	12-node cylindrical hybrid brick
CCL18	18-node cylindrical prism
CCL18H	18-node cylindrical hybrid prism
CCL24	24-node cylindrical brick
CCL24H	24-node cylindrical hybrid brick

CCL24R	24-node cylindrical brick with reduced integration
CCL24RH	24-node cylindrical hybrid brick with reduced integration
CGAX3	3-node generalized linear axisymmetric triangle, twist
CGAX3H	3-node generalized linear axisymmetric triangle, hybrid, constant pressure, twist
CGAX3HT	3-node generalized axisymmetric thermally coupled triangle, hybrid, constant pressure, linear displacement and temperature, twist
CGAX3T	3-node generalized axisymmetric thermally coupled triangle, linear displacement and temperature, twist
CGAX4	4-node generalized bilinear axisymmetric quadrilateral, twist
CGAX4H	4-node generalized bilinear axisymmetric quadrilateral, hybrid, constant pressure, twist
CGAX4HT	4-node generalized axisymmetric thermally coupled quadrilateral, hybrid, constant pressure, bilinear displacement and temperature, twist
CGAX4R	4-node generalized bilinear axisymmetric quadrilateral, reduced integration, hourglass control, twist
CGAX4RH	4-node generalized bilinear axisymmetric quadrilateral, hybrid, constant pressure, reduced integration, hourglass control, twist
CGAX4RHT	4-node generalized axisymmetric thermally coupled quadrilateral, bilinear displacement and temperature, hybrid, constant pressure, reduced integration, hourglass control, twist
CGAX4RT	4-node generalized axisymmetric thermally coupled quadrilateral, bilinear displacement and temperature, reduced integration, hourglass control, twist
CGAX4T	4-node generalized axisymmetric thermally coupled quadrilateral, bilinear displacement and temperature, twist
CGAX6	6-node generalized quadratic axisymmetric triangle, twist
CGAX6H	6-node generalized quadratic axisymmetric triangle, hybrid, linear pressure, twist
CGAX6M	6-node generalized modified axisymmetric triangle, twist, hourglass control
CGAX6MH	6-node generalized modified axisymmetric triangle, twist, hybrid, linear pressure, hourglass control

CGAX6MHT 6-node generalized modified thermally coupled axisymmetric triangle, quadratic displacement, linear temperature, hybrid, linear pressure, twist, hourglass control

CGAX6MT 6-node generalized modified thermally coupled axisymmetric triangle, quadratic displacement, linear temperature, twist, hourglass control

CGAX8 8-node generalized biquadratic axisymmetric quadrilateral, twist

CGAX8H 8-node generalized biquadratic axisymmetric quadrilateral, hybrid, linear pressure, twist

CGAX8HT 8-node generalized axisymmetric thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, hybrid, linear pressure, twist

CGAX8R 8-node generalized biquadratic axisymmetric quadrilateral, reduced integration, twist

CGAX8RH 8-node generalized biquadratic axisymmetric quadrilateral, hybrid, linear pressure, reduced integration, twist

CGAX8RHT 8-node generalized axisymmetric thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, hybrid, linear pressure, reduced integration, twist

CGAX8RT 8-node generalized axisymmetric thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, reduced integration, twist

CGAX8T 8-node generalized axisymmetric thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, twist

CIN3D8 8-node linear one-way infinite brick

CIN3D12R 12-node quadratic one-way infinite brick

CIN3D18R 18-node quadratic one-way infinite brick

CINAX4 4-node linear axisymmetric one-way infinite quadrilateral

CINAX5R 5-node quadratic axisymmetric one-way infinite quadrilateral

CINPE4 4-node linear plane strain one-way infinite quadrilateral

CINPE5R 5-node quadratic plane strain one-way infinite quadrilateral

CINPS4 4-node linear plane stress one-way infinite quadrilateral

CINPS5R 5-node quadratic plane stress one-way infinite quadrilateral

COD2D4C 6-node displacement, pore pressure and slurry concentration cohesive element

COD2D4P 6-node two-dimensional displacement and pore pressure cohesive element with the transition from Darcy flow to Poiseuille flow

COD2D4Q 6-node temperature and slurry concentration cohesive element

COD2D4PT 6-node two-dimensional temperature and pore pressure cohesive element

COD3D6C 9-node displacement, pore pressure and slurry concentration cohesive element

COD3D6P 9-node three-dimensional displacement and pore pressure cohesive element with the transition from Darcy flow to Poiseuille flow

COD3D6Q 9-node temperature and slurry concentration cohesive element

COD3D6PT 9-node three-dimensional temperature and pore pressure cohesive element with the transition from Darcy flow to Poiseuille flow

COD3D8C 12-node displacement, pore pressure, and slurry concentration cohesive element

COD3D8P 12-node three-dimensional displacement and pore pressure cohesive element with the transition from Darcy flow to Poiseuille flow

COD3D8Q 12-node temperature and slurry concentration cohesive element

COD3D8PT 12-node three-dimensional temperature and pore pressure cohesive element with the transition from Darcy flow to Poiseuille flow

CODAX4C 6-node displacement, pore pressure, and slurry concentration cohesive element

CODAX4P 6-node axisymmetric displacement and pore pressure cohesive element with the transition from Darcy flow to Poiseuille flow

CODAX4Q 6-node coupled temperature-slurry concentration cohesive element

CODAX4PT 6-node axisymmetric temperature and pore pressure cohesive element with the transition from Darcy flow to Poiseuille flow

COH2D4 4-node two-dimensional cohesive element

COH2D4P 6-node two-dimensional displacement and pore pressure cohesive element

COH2D4T 4-node two-dimensional displacement and temperature cohesive element

COH3D6 6-node three-dimensional cohesive element

COH3D6P 9-node three-dimensional displacement and pore pressure cohesive element

COH3D6T 6-node three-dimensional displacement and temperature cohesive element

COH3D8	8-node three-dimensional cohesive element
COH3D8P	12-node three-dimensional displacement and pore pressure cohesive element
COH3D8T	8-node three-dimensional displacement and temperature cohesive element
COHAX4	4-node axisymmetric cohesive element
COHAX4P	6-node axisymmetric displacement and pore pressure cohesive element
COHAX4T	4-node axisymmetric displacement and temperature cohesive element
CONN2D2	Connector element in a plane between two nodes or ground and a node
CONN3D2	Connector element in space between two nodes or ground and a node
CPE3	3-node linear plane strain triangle
CPE3E	3-node linear plane strain piezoelectric triangle
CPE3H	3-node linear plane strain triangle, hybrid, constant pressure
CPE3T	3-node plane strain thermally coupled triangle, linear displacement and temperature
CPE4	4-node bilinear plane strain quadrilateral
CPE4E	4-node bilinear plane strain piezoelectric quadrilateral
CPE4H	4-node bilinear plane strain quadrilateral, hybrid, constant pressure
CPE4HT	4-node plane strain thermally coupled quadrilateral, bilinear displacement and temperature, hybrid, constant pressure
CPE4I	4-node bilinear plane strain quadrilateral, incompatible modes
CPE4IH	4-node bilinear plane strain quadrilateral, hybrid, linear pressure, incompatible modes
CPE4P	4-node plane strain quadrilateral, bilinear displacement, bilinear pore pressure
CPE4PH	4-node plane strain quadrilateral, bilinear displacement, bilinear pore pressure, hybrid, constant pressure
CPE4R	4-node bilinear plane strain quadrilateral, reduced integration, hourglass control
CPE4RH	4-node bilinear plane strain quadrilateral, hybrid, constant pressure, reduced integration, hourglass control

CPE4RHT 4-node bilinear plane strain thermally coupled quadrilateral, hybrid, constant pressure, reduced integration, hourglass control

CPE4RP 4-node plane strain quadrilateral, bilinear displacement, bilinear pore pressure, reduced integration, hourglass control

CPE4RPH 4-node plane strain quadrilateral, bilinear displacement, bilinear pore pressure, hybrid, constant pressure, reduced integration, hourglass control

CPE4RT 4-node bilinear plane strain thermally coupled quadrilateral, bilinear displacement and temperature, reduced integration, hourglass control

CPE4T 4-node plane strain thermally coupled quadrilateral, bilinear displacement and temperature

CPE6 6-node quadratic plane strain triangle

CPE6E 6-node quadratic plane strain piezoelectric triangle

CPE6H 6-node quadratic plane strain triangle, hybrid, linear pressure

CPE6M 6-node modified plane strain triangle, hourglass control

CPE6MH 6-node modified quadratic plane strain triangle, hybrid, linear pressure, hourglass control

CPE6MHT 6-node modified quadratic plane strain thermally coupled triangle, hybrid, linear pressure, hourglass control

CPE6MP 6-node modified displacement and pore pressure plane strain triangle, hourglass control

CPE6MPH 6-node modified displacement and pore pressure plane strain triangle, hybrid, linear pressure, hourglass control

CPE6MT 6-node modified quadratic plane strain thermally coupled triangle, hourglass control modified plane strain thermally coupled triangle, hourglass control

CPE8 8-node biquadratic plane strain quadrilateral

CPE8E 8-node biquadratic plane strain piezoelectric quadrilateral

CPE8H 8-node biquadratic plane strain quadrilateral, hybrid, linear pressure

CPE8HT 8-node plane strain thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, hybrid, linear pressure

CPE8P 8-node plane strain quadrilateral, biquadratic displacement, bilinear pore pressure

CPE8PH	8-node plane strain quadrilateral, biquadratic displacement, bilinear pore pressure, hybrid, linear pressure stress
CPE8R	8-node biquadratic plane strain quadrilateral, reduced integration
CPE8RE	8-node biquadratic plane strain piezoelectric quadrilateral, reduced integration
CPE8RH	8-node biquadratic plane strain quadrilateral, hybrid, linear pressure, reduced integration
CPE8RHT	8-node plane strain thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, reduced integration, hybrid, linear pressure
CPE8RP	8-node plane strain quadrilateral, biquadratic displacement, bilinear pore pressure, reduced integration
CPE8RPH	8-node biquadratic displacement, bilinear pore pressure, reduced integration, hybrid, linear pressure
CPE8RT	8-node plane strain thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, reduced integration
CPE8T	8-node plane strain thermally coupled quadrilateral, biquadratic displacement, bilinear temperature
CPEG3	3-node linear generalized plane strain triangle
CPEG3H	3-node linear generalized plane strain triangle, hybrid, constant pressure
CPEG3HT	3-node generalized plane strain thermally coupled triangle, linear displacement and temperature, hybrid, constant pressure
CPEG3T	3-node generalized plane strain thermally coupled triangle, linear displacement and temperature
CPEG4	4-node bilinear generalized plane strain quadrilateral
CPEG4H	4-node bilinear generalized plane strain quadrilateral, hybrid, constant pressure
CPEG4HT	4-node generalized plane strain thermally coupled quadrilateral, bilinear displacement and temperature, hybrid, constant pressure
CPEG4I	4-node bilinear generalized plane strain quadrilateral, incompatible modes
CPEG4IH	4-node bilinear generalized plane strain quadrilateral, hybrid, linear pressure, incompatible modes

CPEG4R	4-node bilinear generalized plane strain quadrilateral, reduced integration, hourglass control
CPEG4RH	4-node bilinear generalized plane strain quadrilateral, hybrid, constant pressure, reduced integration, hourglass control
CPEG4RHT	4-node generalized plane strain thermally coupled quadrilateral, bilinear displacement and temperature, hybrid, constant pressure, reduced integration, hourglass control
CPEG4RT	4-node generalized plane strain thermally coupled quadrilateral, bilinear displacement and temperature, reduced integration, hourglass control
CPEG4T	4-node generalized plane strain thermally coupled quadrilateral, bilinear displacement and temperature
CPEG6	6-node quadratic generalized plane strain triangle
CPEG6H	6-node quadratic generalized plane strain triangle, hybrid, linear pressure
CPEG6M	6-node modified generalized plane strain triangle, hourglass control
CPEG6MH	6-node modified generalized plane strain triangle, hybrid, linear pressure, hourglass control
CPEG6MHT	6-node modified generalized plane strain thermally coupled triangle, quadratic displacement, linear temperature, hybrid, constant pressure, hourglass control
CPEG6MT	6-node modified generalized plane strain thermally coupled triangle, quadratic displacement, linear temperature, hourglass control
CPEG8	8-node biquadratic generalized plane strain quadrilateral
CPEG8H	8-node biquadratic generalized plane strain quadrilateral, hybrid, linear pressure
CPEG8HT	8-node generalized plane strain thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, hybrid, linear pressure
CPEG8R	8-node biquadratic generalized plane strain quadrilateral, reduced integration
CPEG8RH	8-node biquadratic generalized plane strain quadrilateral, hybrid, linear pressure, reduced integration
CPEG8RHT	8-node generalized plane strain thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, hybrid, linear pressure, reduced integration
CPEG8T	8-node generalized plane strain thermally coupled quadrilateral, biquadratic displacement, bilinear temperature

CPS3	3-node linear plane stress triangle
CPS3E	3-node linear plane stress piezoelectric triangle
CPS3T	3-node plane stress thermally coupled triangle, linear displacement and temperature
CPS4	4-node bilinear plane stress quadrilateral
CPS4E	4-node bilinear plane stress piezoelectric quadrilateral
CPS4I	4-node bilinear plane stress quadrilateral, incompatible modes
CPS4R	4-node bilinear plane stress quadrilateral, reduced integration, hourglass control
CPS4RT	4-node plane stress thermally coupled quadrilateral, bilinear displacement and temperature, reduced integration, hourglass control
CPS4T	4-node plane stress thermally coupled quadrilateral, bilinear displacement and temperature
CPS6	6-node quadratic plane stress triangle
CPS6E	6-node quadratic plane stress piezoelectric triangle
CPS6M	6-node modified plane stress triangle, hourglass control
CPS6MT	6-node modified plane stress thermally coupled triangle, hourglass control
CPS8	8-node biquadratic plane stress quadrilateral
CPS8E	8-node biquadratic plane stress piezoelectric quadrilateral
CPS8R	8-node biquadratic plane stress quadrilateral, reduced integration
CPS8RE	8-node biquadratic plane stress piezoelectric quadrilateral, reduced integration
CPS8RT	8-node plane stress thermally coupled quadrilateral, biquadratic displacement, bilinear temperature, reduced integration
CPS8T	8-node plane stress thermally coupled quadrilateral, biquadratic displacement, bilinear temperature
CSS8	8-node linear solid shell brick, incompatible modes, with assumed strain
DASHPOT1	Dashpot between a node and ground, acting in a fixed direction
DASHPOT2	Dashpot between two nodes, acting in a fixed direction

DASHPOTA	Axial dashpot between two nodes, whose line of action is the line joining the two nodes
DC1D2	2-node heat transfer link
DC1D2E	2-node coupled thermal-electrical link
DC1D3	3-node heat transfer link
DC1D3E	3-node coupled thermal-electrical link
DC2D3	3-node linear heat transfer triangle
DC2D3E	3-node linear coupled thermal-electrical triangle
DC2D4	4-node linear heat transfer quadrilateral
DC2D4E	4-node linear coupled thermal-electrical quadrilateral
DC2D6	6-node quadratic heat transfer triangle
DC2D6E	6-node quadratic coupled thermal-electrical triangle
DC2D8	8-node quadratic heat transfer quadrilateral
DC2D8E	8-node quadratic coupled thermal-electrical quadrilateral
DC3D4	4-node linear heat transfer tetrahedron
DC3D4E	4-node linear coupled thermal-electrical tetrahedron
DC3D5	5-node linear heat transfer pyramid
DC3D6	6-node linear heat transfer triangular prism
DC3D6E	6-node linear coupled thermal-electrical triangular prism
DC3D8	8-node linear heat transfer brick
DC3D8E	8-node linear coupled thermal-electrical brick
DC3D8R	8-node linear brick, reduced integration, hourglass control
DC3D10	10-node quadratic heat transfer tetrahedron
DC3D10E	10-node quadratic coupled thermal-electrical tetrahedron

DC3D15	15-node quadratic heat transfer triangular prism
DC3D15E	15-node quadratic coupled thermal-electrical triangular prism
DC3D20	20-node quadratic heat transfer brick
DC3D20E	20-node quadratic coupled thermal-electrical brick
DCAX3	3-node linear axisymmetric heat transfer triangle
DCAX3E	3-node linear axisymmetric coupled thermal-electrical triangle
DCAX4	4-node linear axisymmetric heat transfer quadrilateral
DCAX4E	4-node linear axisymmetric coupled thermal-electrical quadrilateral
DCAX6	6-node quadratic axisymmetric heat transfer triangle
DCAX6E	6-node quadratic axisymmetric coupled thermal-electrical triangle
DCAX8	8-node quadratic axisymmetric heat transfer quadrilateral
DCAX8E	8-node quadratic axisymmetric coupled thermal-electrical quadrilateral
DCC1D2	2-node convection/diffusion link
DCC1D2D	2-node convection/diffusion link, dispersion control
DCC2D4	4-node convection/diffusion quadrilateral
DCC2D4D	4-node convection/diffusion quadrilateral, dispersion control
DCC3D8	8-node convection/diffusion brick
DCC3D8D	8-node convection/diffusion brick, dispersion control
DCCAX2	2-node axisymmetric convection/diffusion link
DCCAX2D	2-node axisymmetric convection/diffusion link, dispersion control
DCCAX4	4-node axisymmetric convection/diffusion quadrilateral
DCCAX4D	4-node axisymmetric convection/diffusion quadrilateral, dispersion control
DCOUP2D	Two-dimensional distributing coupling element
DCOUP3D	Three-dimensional distributing coupling element

DGAP	Unidirectional thermal interactions between two nodes
DRAG2D	2D drag chain, for use in cases where only horizontal motion is being studied
DRAG3D	3D drag chain
DS3	3-node heat transfer triangular shell
DS4	4-node heat transfer quadrilateral shell
DS6	6-node heat transfer triangular shell
DS8	8-node heat transfer quadrilateral shell
DSAX1	2-node axisymmetric heat transfer shell
DSAX2	3-node axisymmetric heat transfer shell
ELBOW31	2-node pipe in space with deforming section, linear interpolation along the pipe
ELBOW31B	2-node pipe in space with ovalization only, axial gradients of ovalization neglected
ELBOW31C	2-node pipe in space with ovalization only, axial gradients of ovalization neglected. This is the same as element type ELBOW31B except that the odd numbered terms in the Fourier interpolation around the pipe, except the first term, are neglected.
ELBOW32	3-node pipe in space with deforming section, quadratic interpolation along the pipe
EMC2D3	3-node triangular zero-order electromagnetic element
EMC2D4	4-node quadrilateral zero-order electromagnetic element
EMC3D4	4-node tetrahedral zero-order electromagnetic element
EMC3D6	6-node prism zero-order electromagnetic element
EMC3D8	8-node hexahedral zero-order electromagnetic element
FP2D2	2-node two-dimensional linear fluid pipe element
FP2D2AQ	8-node linear, annular two-dimensional coupled temperature-slurry-pore pressure fluid pipe element
FP2D2AT	8-node two-dimensional annular coupled temperature–pore pressure fluid pipe element
FP2D2C	2-node linear two-dimensional coupled slurry-pore pressure fluid pipe element

FP2D2Q	6-node linear two-dimensional coupled temperature-slurry-pore pressure fluid pipe element
FP2D2T	6-node two-dimensional coupled temperature–pore pressure fluid pipe element
FP3D2	2-node three-dimensional linear fluid pipe element
FP3D2AT	8-node three-dimensional annular coupled temperature–pore pressure fluid pipe element
FP3D2T	6-node three-dimensional coupled temperature–pore pressure fluid pipe element
FPC2D2	2-node two-dimensional linear fluid pipe connector element
FPC2D2AT	8-node two-dimensional annular coupled temperature–pore pressure fluid pipe connector element
FPC2D2C	2-node linear two-dimensional coupled slurry-pore pressure fluid pipe connector element
FPC2D2AQ	8-node linear, annular two-dimensional coupled temperature-surry pore pressure fluid pipe connector element
FPC2D2Q	6-node linear two-dimensional coupled temperature-surry pore pressure fluid pipe connector element
FPC2D2T	6-node two-dimensional coupled temperature–pore pressure fluid pipe connector element
FPC3D2	2-node three-dimensional linear fluid pipe connector element
FPC3D2AQ	8-node linear, annular three-dimensional coupled temperature-slurry-pore pressure fluid pipe connector element
FPC3D2AT	8-node three-dimensional annular coupled temperature–pore pressure fluid pipe connector element
FPC3D2C	2-node linear three-dimensional coupled slurry-pore pressure fluid pipe connector element
FPC3D2Q	6-node linear three-dimensional coupled temperature-slurry-pore pressure fluid pipe connector element
FPC3D2T	6-node three-dimensional coupled temperature–pore pressure fluid pipe connector element
FRAME2D	2-node two-dimensional straight frame element

FRAME3D	2-node three-dimensional straight frame element
GAPCYL	Cylindrical gap between two nodes
GAPSPHER	Spherical gap between two nodes
GAPUNI	Unidirectional gap between two nodes
GAPUNIT	Unidirectional gap and thermal interactions between two nodes
GK2D2	2-node two-dimensional gasket element
GK2D2N	2-node two-dimensional gasket element with thickness-direction behavior only
GK3D2	2-node three-dimensional gasket element
GK3D2N	2-node three-dimensional gasket element with thickness-direction behavior only
GK3D4L	4-node three-dimensional line gasket element
GK3D4LN	4-node three-dimensional line gasket element with thickness-direction behavior only
GK3D6L	6-node three-dimensional line gasket element
GK3D6LN	6-node three-dimensional line gasket element with thickness-direction behavior only
GK3D6	6-node three-dimensional gasket element
GK3D6N	6-node three-dimensional gasket element with thickness-direction behavior only
GK3D6T	6-node three-dimensional displacement and temperature gasket element
GK3D6TN	6-node three-dimensional displacement and temperature gasket element with thickness-direction behavior only
GK3D8	8-node three-dimensional gasket element
GK3D8N	8-node three-dimensional gasket element with thickness-direction behavior only
GK3D8T	8-node three-dimensional displacement and temperature gasket element
GK3D8TN	8-node three-dimensional displacement and temperature gasket element with thickness-direction behavior only
GK3D12M	12-node three-dimensional gasket element
GK3D12MN	12-node three-dimensional gasket element with thickness-direction behavior only

GK3D18	18-node three-dimensional gasket element
GK3D18N	18-node three-dimensional gasket element with thickness-direction behavior only
GKAX2	2-node axisymmetric gasket element
GKAX2N	2-node axisymmetric gasket element with thickness-direction behavior only
GKAX4	4-node axisymmetric gasket element
GKAX4N	4-node axisymmetric gasket element with thickness-direction behavior only
GKAX6	6-node axisymmetric gasket element
GKAX6N	6-node axisymmetric gasket element with thickness-direction behavior only
GKPE4	4-node plane strain gasket element
GKPE6	6-node plane strain gasket element
GKPS4	4-node plane stress gasket element
GKPS4N	4-node two-dimensional gasket element with thickness-direction behavior only
GKPS6	6-node plane stress gasket element
GKPS6N	6-node two-dimensional gasket element with thickness-direction behavior only
HEATCAP	Point heat capacitance
IRS21A	Axisymmetric rigid surface element (for use with first-order axisymmetric elements)
IRS22A	Axisymmetric rigid surface element (for use with second-order axisymmetric elements)
ISL21A	2-node axisymmetric slide line element (for use with first-order axisymmetric elements)
ISL22A	3-node axisymmetric slide line element (for use with second-order axisymmetric elements)
ITSCYL	Cylindrical geometry tube support interaction element
ITSUNI	Unidirectional tube support interaction element
ITT21	Tube-tube element for use with first-order, 2D beam and pipe elements
ITT31	Tube-tube element for use with first-order, 3D beam and pipe elements

JOINT2D	Two-dimensional elastic-plastic joint interaction element. These elements are available only for use in Abaqus/Aqua.
JOINT3D	Three-dimensional elastic-plastic joint interaction element. These elements are available only for use in Abaqus/Aqua.
JOINTC	Three-dimensional joint interaction element
LS3S	3-node second-order line spring for use on a symmetry plane
LS6	6-node general second-order line spring. This element can be used only with linear elastic material behavior.
M3D3	3-node triangular membrane
M3D4	4-node quadrilateral membrane
M3D4R	4-node quadrilateral membrane, reduced integration, hourglass control
M3D6	6-node triangular membrane
M3D8	8-node quadrilateral membrane
M3D8R	8-node quadrilateral membrane, reduced integration
M3D9	9-node quadrilateral membrane
M3D9R	9-node quadrilateral membrane, reduced integration, hourglass control
MASS	Point mass
MAX1	2-node linear axisymmetric membrane
MAX2	3-node quadratic axisymmetric membrane
MCL6	6-node cylindrical membrane
MCL9	9-node cylindrical membrane
MGAX1	2-node linear axisymmetric membrane, twist
MGAX2	3-node quadratic axisymmetric membrane, twist
PIPE21	2-node linear pipe in a plane
PIPE21H	2-node linear pipe in a plane, hybrid formulation

PIPE22	3-node quadratic pipe in a plane
PIPE22H	3-node quadratic pipe in a plane, hybrid formulation
PIPE31	2-node linear pipe in space
PIPE31H	2-node linear pipe in space, hybrid formulation
PIPE32	3-node quadratic pipe in space
PIPE32H	3-node quadratic pipe in space, hybrid formulation
PSI24	4-node 2D pipe-soil interaction element
PSI26	6-node 2D pipe-soil interaction element
PSI34	4-node 3D pipe-soil interaction element
PSI36	6-node 3D pipe-soil interaction element
Q3D4	4-node tetrahedron, linear displacement, linear electric potential and linear temperature
Q3D6	6-node linear triangular prism, linear displacement, linear electric potential and linear temperature
Q3D8	8-node brick, trilinear displacement, trilinear electric potential and trilinear temperature
Q3D8H	8-node brick, trilinear displacement, trilinear electric potential, trilinear temperature, hybrid, constant pressure
Q3D8R	8-node brick, trilinear displacement, trilinear electric potential, trilinear temperature, reduced integration, hourglass control
Q3D8RH	8-node brick, trilinear displacement, trilinear electric potential, trilinear temperature, reduced integration, hourglass control, hybrid, constant pressure
Q3D10M	10-node modified displacement, electric potential, temperature quadratic tetrahedron, hourglass control
Q3D10MH	10-node modified displacement, electric potential, temperature quadratic tetrahedron, hybrid, linear pressure, hourglass control
Q3D20	20-node quadratic brick, triquadratic displacement, trilinear electric potential, trilinear temperature
Q3D20H	20-node quadratic brick, triquadratic displacement, trilinear electric potential, trilinear temperature, hybrid, linear pressure

Q3D20R	20-node quadratic brick, triquadratic displacement, trilinear electric potential, trilinear temperature, reduced integration
Q3D20RH	20-node quadratic brick, triquadratic displacement, trilinear electric potential, trilinear temperature, hybrid, linear pressure, reduced integration
QAX3	3-node linear displacement, electric potential, and temperature
QAX4	4-node bilinear displacement, electric potential, and temperature
QAX6M	6-node modified displacement, electric potential, and temperature, with hourglass control
QAX8	8-node biquadratic displacement, bilinear electric potential, and bilinear temperature
QEC3D4	4-node linear electric potential in solid, temperature, electric potential in electrolyte, ion concentration; optionally displacement, and either pore pressure or species concentration
QEC3D6	6-node linear electric potential in solid, temperature, electric potential in electrolyte, ion concentration; optionally displacement, and either pore pressure or species concentration
QEC3D8	8-node trilinear displacement, electric potential in solid, temperature, electric potential in electrolyte, ion concentration, species concentration
QEC3D8R	8-node trilinear displacement, electric potential in solid, temperature, electric potential in electrolyte, ion concentration, species concentration; reduced integration with hourglass control
R2D2	2-node 2D linear rigid link (for use in plane strain or plane stress)
R3D3	3-node 3D rigid triangular facet
R3D4	4-node 3D bilinear rigid quadrilateral
RAX2	2-node linear axisymmetric rigid link (for use in axisymmetric planar geometries)
RB2D2	2-node 2D rigid beam
RB3D2	2-node 3D rigid beam
ROTARYI	Rotary inertia at a point
S3	3-node triangular general-purpose shell, finite membrane strains (identical to element S3R)

S3T 3-node thermally coupled triangular general-purpose shell, finite membrane strains (identical to element S3RT)

S3R 3-node triangular general-purpose shell, finite membrane strains (identical to element S3)

S3RT 3-node thermally coupled triangular general-purpose shell, finite membrane strains (identical to element S3T)

S4 4-node general-purpose shell, finite membrane strains

S4T 4-node thermally coupled general-purpose shell, finite membrane strains

S4R 4-node general-purpose shell, reduced integration, hourglass control, finite membrane strains

S4RT 4-node thermally coupled general-purpose shell, reduced integration, hourglass control, finite membrane strains

S4R5 4-node thin shell, reduced integration, hourglass control, using five degrees of freedom per node

S8R 8-node doubly curved thick shell, reduced integration

S8R5 8-node doubly curved thin shell, reduced integration, using five degrees of freedom per node

S8RT 8-node thermally coupled quadrilateral general thick shell, biquadratic displacement, bilinear temperature in the shell surface

S9R5 9-node doubly curved thin shell, reduced integration, using five degrees of freedom per node

SAX1 2-node linear axisymmetric thin or thick shell

SAX2 3-node quadratic axisymmetric thin or thick shell

SAX2T 3-node axisymmetric thermally coupled thin or thick shell, quadratic displacement, linear temperature in the shell surface

SAXA1N Linear asymmetric-axisymmetric, Fourier shell element with 2 nodes in the generator direction and *N* Fourier modes

SAXA2N Quadratic asymmetric-axisymmetric, Fourier shell element with 3 nodes in the generator direction and *N* Fourier modes

SC6R 6-node triangular in-plane continuum shell wedge, general-purpose continuum shell, finite membrane strains.

SC8R	8-node quadrilateral in-plane general-purpose continuum shell, reduced integration with hourglass control, finite membrane strains.
SC6RT	6-node linear displacement and temperature, triangular in-plane continuum shell wedge, general-purpose continuum shell, finite membrane strains
SC8RT	8-node linear displacement and temperature, hexahedron, general-purpose continuum shell, finite membrane strains
SFM3D3	3-node triangular surface element
SFM3D4	4-node quadrilateral surface element
SFM3D4R	4-node quadrilateral surface element, reduced integration
SFM3D6	6-node triangular surface element
SFM3D8	8-node quadrilateral surface element
SFM3D8R	8-node quadrilateral surface element, reduced integration
SFMAX1	2-node linear axisymmetric surface element
SFMAX2	3-node quadratic axisymmetric surface element
SFMCL6	6-node cylindrical surface element
SFMCL9	9-node cylindrical surface element
SFMGAX1	2-node linear axisymmetric surface element, twist
SFMGAX2	3-node quadratic axisymmetric surface element, twist
SHEAR4	4-node shear panel element
SPRING1	Spring between a node and ground, acting in a fixed direction
SPRING2	Spring between two nodes, acting in a fixed direction
SPRINGA	Axial spring between two nodes, whose line of action is the line joining the two nodes. This line of action may rotate in large-displacement analysis
STRI3	3-node triangular facet thin shell
STRI65	6-node triangular thin shell, using five degrees of freedom per node

SUBSTR	Generic substructure element. The preferred method for including substructures in a model is to associate all elements with the generic substructure element type SUBSTR.
T2D2	2-node linear 2D truss
T2D2E	2-node 2D piezoelectric truss
T2D2H	2-node linear 2D truss, hybrid
T2D2T	2-node 2D thermally coupled truss
T2D3	3-node quadratic 2D truss
T2D3E	3-node 2D piezoelectric truss
T2D3H	3-node quadratic 2D truss, hybrid
T2D3T	3-node 2D thermally coupled truss
T3D2	2-node linear 3D truss
T3D2E	2-node 3D piezoelectric truss
T3D2H	2-node linear 3D truss, hybrid
T3D2T	2-node 3D thermally coupled truss
T3D3	3-node quadratic 3D truss
T3D3E	3-node 3D piezoelectric truss
T3D3H	3-node quadratic 3D truss, hybrid
T3D3T	3-node 3D thermally coupled truss
WARP2D3	3-node linear 2D warping element
WARP2D4	4-node bilinear 2D warping element
WARPF2D3	3-node linear 2D warping element with free (unconstrained) in-plane and out-of-plane warping
WARPF2D4	4-node bilinear 2D warping element with free (unconstrained) in-plane and out-of-plane warping
WARPF2D6	6-node quadratic 2D warping element with free (unconstrained) in-plane and out-of-plane warping

WARPF2D8	8-node quadratic 2D warping element with free (unconstrained) in-plane and out-of-plane warping

Abaqus/Explicit Element Index

This index provides a reference to all of the element types that are available in Abaqus/Explicit.

Elements are listed in alphabetical order, where numerical characters precede the letter "A" and two-digit numbers are put in numerical, rather than "alphabetical," order. For example, C3D8R precedes CAX3.

For certain options, such as contact and surface-based distributing coupling, Abaqus may generate internal elements (such as IDCOUP3D for surface-based distributing coupling). These internal element names are not included in the index below but may appear in an output database (`.odb`) or data (`.dat`) file.

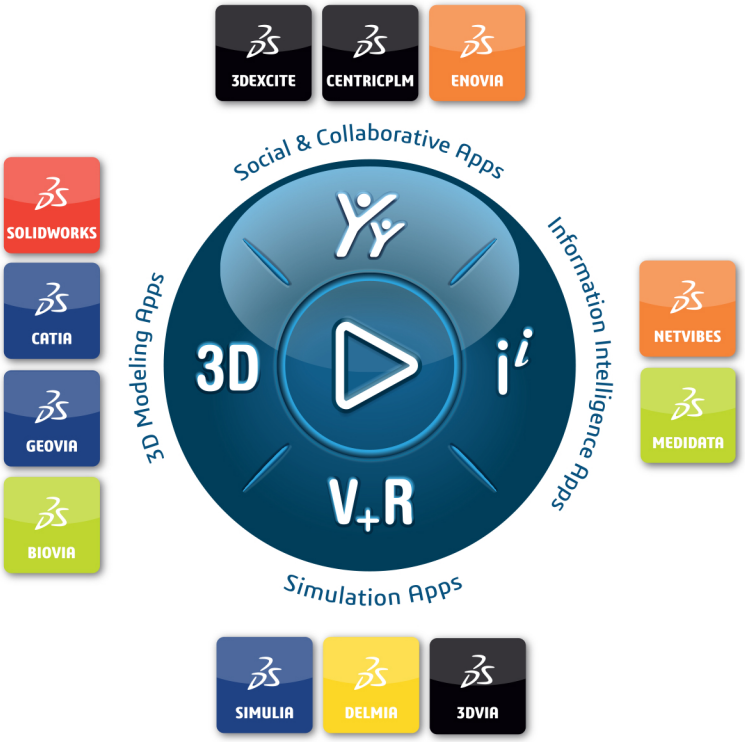
AC2D3	3-node linear 2D acoustic triangle
AC2D4R	4-node linear 2D acoustic quadrilateral, reduced integration, hourglass control
AC3D4	4-node linear acoustic tetrahedron
AC3D6	6-node linear acoustic triangular prism
AC3D8R	8-node linear acoustic brick, reduced integration, hourglass control
ACAX3	3-node linear axisymmetric acoustic triangle
ACAX4R	4-node linear axisymmetric acoustic quadrilateral, reduced integration, hourglass control
ACIN2D2	2-node linear 2D acoustic infinite element
ACIN3D3	3-node linear 3D acoustic infinite element
ACIN3D4	4-node linear 3D acoustic infinite element
ACINAX2	2-node linear axisymmetric acoustic infinite element
B21	2-node linear beam in a plane
B22	3-node quadratic beam in a plane
B31	2-node linear beam in space
B32	3-node quadratic beam in space
C3D4	4-node linear tetrahedron
C3D4H	4-node linear tetrahedron, hybrid, linear pressure
C3D5	5-node linear pyramid

C3D4T	4-node thermally coupled tetrahedron, linear displacement and temperature
C3D6	6-node linear triangular prism, reduced integration, hourglass control
C3D6T	6-node thermally coupled triangular prism, linear displacement and temperature, reduced integration, hourglass control
C3D8	8-node linear brick
C3D8I	8-node linear brick, incompatible modes
C3D8IT	8-node linear brick, displacement and temperature, incompatible modes
C3D8R	8-node linear brick, reduced integration, hourglass control
C3D8T	8-node thermally coupled brick, trilinear displacement and temperature
C3D8RT	8-node thermally coupled brick, trilinear displacement and temperature, reduced integration, hourglass control
C3D10	10-node quadratic tetrahedron
C3D10M	10-node modified tetrahedron, hourglass control
C3D10MT	10-node modified thermally coupled tetrahedron, hourglass control
C3D10T	10-node triquadratic displacement, triquadratic temperature
CAX3	3-node linear axisymmetric triangle
CAX3T	3-node axisymmetric thermally coupled triangle, linear displacement and temperature
CAX4R	4-node bilinear axisymmetric quadrilateral, reduced integration, hourglass control
CAX4RT	4-node thermally coupled axisymmetric quadrilateral, bilinear displacement and temperature, hybrid, constant pressure, reduced integration, hourglass control
CAX6M	6-node modified axisymmetric triangle, hourglass control
CAX6MT	6-node modified axisymmetric thermally coupled triangle, hourglass control
CIN3D8	8-node linear one-way infinite brick
CINAX4	4-node linear axisymmetric one-way infinite quadrilateral
CINPE4	4-node linear plane strain one-way infinite quadrilateral

CINPS4	4-node linear plane stress one-way infinite quadrilateral
COHAX4	4-node axisymmetric cohesive element
COH2D4	4-node two-dimensional cohesive element
COH3D6	6-node three-dimensional cohesive element
COH3D8	8-node three-dimensional cohesive element
CONN2D2	Connector element in a plane between two nodes or ground and a node
CONN3D2	Connector element in space between two nodes or ground and a node
CPE3	3-node linear plane strain triangle
CPE3T	3-node plane strain thermally coupled triangle, linear displacement and temperature
CPE4R	4-node bilinear plane strain quadrilateral, reduced integration, hourglass control
CPE4RT	4-node bilinear plane strain thermally coupled quadrilateral, bilinear displacement and temperature, reduced integration, hourglass control
CPE6M	6-node modified plane strain triangle, hourglass control
CPE6MT	6-node modified plane strain thermally coupled triangle, hourglass control
CPS3	3-node linear plane stress triangle
CPS3T	3-node plane stress thermally coupled triangle, linear displacement and temperature
CPS4R	4-node bilinear plane stress quadrilateral, reduced integration, hourglass control
CPS4RT	4-node plane stress thermally coupled quadrilateral, bilinear displacement and temperature, reduced integration, hourglass control
CPS6M	6-node modified plane stress triangle, hourglass control
CPS6MT	6-node modified plane stress thermally coupled triangle, hourglass control
DASHPOTA	Axial dashpot between two nodes, whose line of action is the line joining the two nodes
EC3D8R	8-node linear multi-material Eulerian brick, reduced integration, hourglass control
EC3D8RT	8-node thermally coupled linear multi-material Eulerian brick, reduced integration, hourglass control

HEATCAP	Point heat capacitance
M3D3	3-node triangular membrane
M3D4	4-node quadrilateral membrane
M3D4R	4-node quadrilateral membrane, reduced integration, hourglass control
MASS	Point mass
PC3D	1-node continuum particle element
PD3D	1-node discrete particle element
PIPE21	2-node linear pipe in a plane
PIPE31	2-node linear pipe in space
R2D2	2-node 2D linear rigid link (for use in plane strain or plane stress)
R3D3	3-node 3D rigid triangular facet
R3D4	4-node 3D bilinear rigid quadrilateral
RAX2	2-node linear axisymmetric rigid link (for use in axisymmetric planar geometries)
ROTARYI	Rotary inertia at a point
S3R	3-node triangular shell, finite membrane strains
S3RS	3-node triangular shell, small membrane strains
S3RT	3-node thermally-coupled triangular general-purpose shell, finite membrane strains
S4	4-node general-purpose shell, finite membrane strains
S4R	4-node general-purpose shell, reduced integration, hourglass control, finite membrane strains
S4RS	4-node shell, reduced integration, hourglass control, small membrane strains
S4RSW	4-node shell, reduced integration, hourglass control, small membrane strains, warping considered in small-strain formulation
S4RT	4-node thermally coupled general-purpose shell, reduced integration, hourglass control, finite membrane strains

SAX1	2-node linear axisymmetric thin or thick shell
SC6R	6-node triangular in-plane continuum shell wedge, general-purpose continuum shell, finite membrane strains.
SC8R	8-node quadrilateral in-plane general-purpose continuum shell, reduced integration with hourglass control, finite membrane strains.
SC6RT	6-node linear displacement and temperature, triangular in-plane continuum shell wedge, general-purpose continuum shell, finite membrane strains
SC8RT	8-node linear displacement and temperature, hexahedron, general-purpose continuum shell, finite membrane strains
SFM3D3	3-node triangular surface element
SFM3D4R	4-node quadrilateral surface element, reduced integration
SPRINGA	Axial spring between two nodes, whose line of action is the line joining the two nodes. This line of action may rotate in large-displacement analysis
T2D2	2-node linear 2D truss
T3D2	2-node linear 3D truss



Our **3D**EXPERIENCE® platform powers our brand applications, serving 12 industries, and provides a rich portfolio of industry solution experiences.

Dassault Systèmes is a catalyst for human progress. We provide business and people with collaborative virtual environments to imagine sustainable innovations. By creating virtual twin experiences of the real world with our **3D**EXPERIENCE platform and applications, our customers can redefine the creation, production and life-cycle-management processes of their offer and thus have a meaningful impact to make the world more sustainable. The beauty of the Experience Economy is that it is a human-centered economy for the benefit of all – consumers, patients and citizens.

Dassault Systèmes brings value to more than 300,000 customers of all sizes, in all industries, in more than 150 countries. For more information, visit **www.3ds.com**.

Europe/Middle East/Africa
Dassault Systèmes
10, rue Marcel Dassault
CS 40501
78946 Vélizy-Villacoublay Cedex
France

Asia-Pacific
Dassault Systèmes
17F, Foxconn Building,
No. 1366, Lujiazui Ring Road
Pilot Free Trade Zone, Shanghai 200120
China

Americas
Dassault Systèmes
175 Wyman Street
Waltham, Massachusetts
02451-1223
USA

DASSAULT SYSTEMES

©2026 Dassault Systèmes. All rights reserved. **3DEXPERIENCE**, the 3DS logo, the Compass icon, IFWE, 3DEXCITE, 3DVIA, BIOVIA, CATIA, CENTRIC PLM, DELMIA, ENOVIA, GEOVIA, MEDIDATA, NETVIBES, OUTSCALE, SIMULIA and SOLIDWORKS are commercial trademarks or registered trademarks of Dassault Systèmes, a European company (Societas Europaea) incorporated under French law, and registered with the Versailles trade and companies registry under number 322 306 440, or its subsidiaries in the United States and/or other countries.