SIMULIA

3DEXPERIENCE

ABAQUS 2026 FD04
PRESCRIBED CONDITIONS GUIDE

DASSAULT SYSTEMES

Contents

Trademarks and Legal Notices....6
Abaqus Prescribed Conditions Guide....7
What's New....8
About Prescribed Conditions....22
Amplitude Variations....22
Applying Boundary Conditions and Loads in a Local Coordinate System....22
Loads and Predefined Fields Available for Various Procedures....23
Amplitude Curves....24
Amplitude Curves....24
Defining the Time Period....25
Continuation of an Amplitude Reference in Subsequent Steps....25
Specifying Relative or Absolute Data....25
Relative Data....25
Absolute Data....26
Defining the Amplitude Data....26
Defining Tabular Data....26
Defining Equally Spaced Data....27
Defining Periodic Data....28
Defining Modulated Data....29
Defining Exponential Decay....29
Defining Smooth Step Data....30
Defining a Solution-Dependent Amplitude for Superplastic Forming Analysis....32
Defining the Bubble Load Amplitude for an Underwater Explosion....33
Defining an Amplitude via a User Subroutine....34
Defining an Actuator Amplitude via Co-Simulation....34
Using an Amplitude Definition with Boundary Conditions....34
Using an Amplitude Definition with Secondary Base Motion in Modal Dynamics....36
Defining Multiple Amplitude Curves....37
Defining a Normalized Amplitude Curve....37
Scaling and Shifting Amplitude Curves....37
Reading the Data from an Alternate File....38
Baseline Correction in Abaqus/Standard....38
Effects of Baseline Correction....38
Initial Conditions....40
Specifying Initial Conditions....40
Reading the Input Data from an External File....40
Importing Data from an Output Database File....41
Consistency with Kinematic Constraints....42
Spatial Interpolation Method....42
Initial Condition Types....44
Defining Initial Acoustic Static Pressure....44
Defining Initial Volume Fraction of Material....44

Defining Initial Normalized Concentration....45
Defining Initially Bonded Contact Surfaces....45
Defining Initial Damage Initiation....46
Defining the Initial Location of an Enriched Feature....47
Defining Initial Values of Element Solution-Dependent Variables....47
Defining Initial Values of Predefined Field Variables....48
Defining Initial Fluid Electric Potential....50
Defining Initial Fluid Pressure in Fluid-Filled Structures....51
Defining Initial Values of State Variables for Plastic Hardening....51
Defining Elements Initially Open for Tangential Fluid Flow....53
Defining Initial Slurry Concentration....53
Defining Initial Ion Concentration....53
Defining Initial Species Concentration....53
Defining Initial Mass Flow Rates in Forced Convection Heat Transfer Elements....54
Defining Initial Maximum Energy Density in Mullins Effect Models....54
Defining Initial Values of Plastic Strain....54
Defining Initial Pore Fluid Pressures in a Porous Medium....55
Defining Initial Pressure Stress in a Mass Diffusion Analysis....57
Defining Initial Void Ratios in a Porous Medium....57
Defining Initial Porosity in a Porous Medium....58
Defining a Reference Mesh for Membrane Elements and Three-Dimensional Solid Elements....59
Defining Initial Relative Density....60
Defining Initial Angular and Translational Velocity....60
Defining Initial Saturation for a Porous Medium....60
Defining Initial Solid Electric Potential....61
Defining the Initial Values of Solution-Dependent State Variables....61
Defining Initial Specific Energy for Equations of State....62
Defining Spud Can Embedment or Spud Can Preload....62
Defining Initial Stresses....62
Defining Elevation-Dependent (Geostatic) Initial Stresses....67
Defining Initial Temperatures....67
Defining the Initial Configuration in a One-Step Inverse Analysis....70
Defining Initial Velocities for Specified Degrees of Freedom....70
Defining Initial Volume Fractions for Eulerian Elements....71

Boundary Conditions....72

Prescribing Boundary Conditions as Model Data....72
Using the Direct Format....73
Using the “Type” Format in Stress/Displacement Analyses....73
Prescribing Boundary Conditions at Phantom Nodes for Enriched Elements....75
Prescribing Boundary Conditions as History Data....76
Using the Direct Format....76
Prescribed Displacement....77
Using the “Type” Format in Stress/Displacement Analyses....77
Prescribing Boundary Conditions at Phantom Nodes for Enriched Elements....78
Defining Boundary Conditions That Vary with Time....78
Defining Boundary Conditions through User Subroutines....79
Boundary Condition Propagation....79
Modifying Boundary Conditions....79

Removing Boundary Conditions....80
Fixing Degrees of Freedom at a Point in an Abaqus/Standard Analysis....80
Prescribing Boundary Conditions in Linear Perturbation Steps....81
Prescribing Real and Imaginary Values in Boundary Conditions....81
Prescribed Motion in Modal Superposition Procedures....81
Submodeling....82
Prescribing Large Rotations....82
Example: Using Velocity-Type Boundary Conditions to Prescribe Rotations....82
Prescribing Radial Motion on an Axisymmetric Model....83
Loads....84
About Loads....85
Element-Based Versus Surface-Based Distributed Loads....85
Varying the Magnitude of a Load....86
Loading during General Analysis Steps....86
Loading during Linear Perturbation Steps....89
Concentrated Loads....90
Concentrated Loads....90
Defining Time-Dependent Concentrated Loads....91
Modifying Concentrated Loads....91
Improving the Rate of Convergence in Large-Displacement Implicit Analysis....91
Distributed Loads....92
Defining Time-Dependent Distributed Loads....93
Modifying Distributed Loads....93
Improving the Rate of Convergence in Large-Displacement Implicit Analysis....93
Defining Distributed Loads in a User Subroutine....93
Specifying the Region to Which a Distributed Load Is Applied....93
Applying Electric Machine Loads....94
Body Forces....95
Surface Tractions and Pressure Loads....100
Edge Tractions and Moments on Shell Elements and Line Loads on Beam Elements....114
References....121
Surface Tension Loads....122
Surface Tension....122
Specifying Surface Tension Loads....122
Examples....125
References....129
Fluid Pressure Penetration Loads....130
Simulating Effects of Fluid Pressure Acting on Surfaces that Consider Contact Conditions....130
Surface Fluid Pressure Penetration Loading with General Contact....134
Pairwise Fluid Pressure Penetration Loading with Contact Pairs in Abaqus/Standard....138
Behavior in Linear Perturbation Steps....142
Output....143
Thermal Loads....144
Modeling Thermal Radiation....145
Prescribing Heat Fluxes Directly....145
Prescribing Boundary Convection....148
Prescribing Boundary Radiation....155
Electromagnetic Loads....161

Defining Time-Dependent Electromagnetic Loads....161
Modifying Electromagnetic Loads....161
Prescribing Electromagnetic Loads for Piezoelectric Analyses....162
Prescribing Electromagnetic Loads for Coupled Thermal-Electrical and Fully Coupled Thermal-Electrical-Structural Analyses....163
Prescribing Electromagnetic Loads for Eddy Current and/or Magnetostatic Analyses....165
Acoustic and Shock Loads....167
Specified Boundary Impedance....167
Radiation Boundaries for Exterior Problems....170
Concentrated Pressure-Conjugate Load....173
Incident Wave Loading due to External Sources....173
Pore Fluid Flow....196
Defining Pore Fluid Flow as a Function of the Current Pore Pressure in Consolidation Analysis....196
Prescribing Seepage Flow Velocity and Seepage Flow Directly in Consolidation Analysis....199
Prescribed Assembly Loads....203
Concept of an Assembly Load....203
Modeling an Assembly Load....204
Modeling a Fastener with Continuum Elements....204
Modeling a Fastener with Truss or Beam Elements....207
Updating the Normal in a Large-Displacement Analysis....208
Defining Multiple Pre-Tension Sections....208
Use with Nodal Transformations....208
Applying the Prescribed Assembly Load....208
Prescribing the Pre-Tension Force....208
Prescribing a Tightening Adjustment....209
Controlling the Pre-Tension Node during the Analysis....209
Display of Results....210
Limitations When Using Assembly Loads....210
Procedures....210
Output....210
Input File Template....211
Predefined Fields....212
Predefined Temperature....213
Restrictions....213
Predefined Field Variables....213
Pore Fluid Pressure....214
Restrictions....215
Predefined Pressure Stress....215
Restrictions....215
Predefined Mass Flow Rate....215
Restrictions....215
Specifying Uniform Predefined Temperatures and Fields....216
Reading Initial Values of a Field from a User-Specified Results File....216
Reading Initial Values of a Temperature Field from a User-Specified Output Database File....216
Initializing Predefined Field Variables from a User-Specified Output Database File in Abaqus/Standard....217
Defining Time-Dependent Fields....217
Field Propagation....217
Modifying Fields....217
Removing Fields....218

Reading the Values of a Field Directly from an Alternate Input File....219
Reading the Values of a Field from a User-Specified File....219
Reading Field Values from a User-Specified Results File....220
Reading Temperature Values from a User-Specified Output Database File....221
Defining Fields Using Nodal Scalar Output Values from a User-Specified Output Database File....222
Interpolating Data between Meshes....222
Specifying the Step and Increment to Be Read from the File....225
Interpolation in Time....226
Restrictions....227
Defining the Values of a Predefined Field in a User Subroutine....228
Updating Multiple Predefined Field Variables....228
Defining Solution-Dependent Field Variables....229
Data Hierarchy....229
Element Type Considerations....229
Use in a Mass Diffusion Analysis....229
Use with Beam and Shell Elements....229
Temperature and Field Variable Compatibility across Elements....230

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 Prescribed Conditions Guide

The Abaqus Prescribed Conditions Guide describes how to model nonzero initial conditions, boundary conditions, loads, and predefined fields.

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.

The following types of external conditions can be prescribed in an Abaqus model:

Initial conditions

Nonzero initial conditions can be defined for many variables, as described in *Initial Conditions*.

Boundary conditions

Boundary conditions are used to prescribe values of basic solution variables: displacements and rotations in stress/displacement analysis, temperature in heat transfer or coupled thermal-stress analysis, electrical potential in coupled thermal-electrical analysis, pore pressure in soils analysis, acoustic pressure in acoustic analysis, etc. Boundary conditions can be defined as described in *Boundary Conditions*.

Loads

Many types of loading are available, depending on the analysis procedure. *About Loads* gives an overview of loading in Abaqus. Load types specific to one analysis procedure are described in the appropriate procedure section in *Analysis Procedures*. General loads, which can be applied in multiple analysis types, are described in:

- *Concentrated Loads*
- *Distributed Loads*
- *Thermal Loads*
- *Electromagnetic Loads*
- *Acoustic and Shock Loads*
- *Pore Fluid Flow*

Prescribed assembly loads

Pre-tension sections can be defined in Abaqus/Standard to prescribe assembly loads in bolts or any other type of fastener. Pre-tension sections are described in *Prescribed Assembly Loads*.

Connector loads and motions

Connector elements can be used to define complex mechanical connections between parts, including actuation with prescribed loads or motions. Connector elements are described in *About Connectors*.

Predefined fields

Predefined fields are time-dependent, non-solution-dependent fields that exist over the spatial domain of the model. Temperature is the most commonly defined field. Predefined fields are described in *Predefined Fields*.

What's New

This page describes recent changes in Abaqus Prescribed Conditions.

2026 FD04

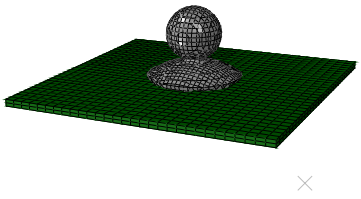
Modeling Surface Tension Loads with Eulerian Elements in Abaqus/Explicit

You can now specify surface tension loads on Eulerian material surfaces in Abaqus/Explicit.

Benefits: You can model large deformation under surface tension loads, including droplet coalescence and droplet break up.

In earlier releases, surface tension loads on element-based surfaces were available only for Lagrangian solid, membrane, and surface elements. Lagrangian elements can get distorted under very large deformations, limiting their ability to model droplet coalescence or break up. Modeling surface tension loads with Eulerian elements is more suitable for such applications.

The animation below shows two fluid droplets with an initial cubic shape, transforming into spherical shapes and ultimately coalescing into a single body under the action of surface tension loads in Abaqus/Explicit. The two fluid droplets have a fixed contact angle and are modeled as a linear equation of state (EOS) material of type USUP using EC3D8R elements. A surface tension load is applied on the Eulerian material surface, which is created automatically on the exterior of a particular Eulerian material instance.



For more information, see **DSLOAD* and *Surface Tension Loads*.

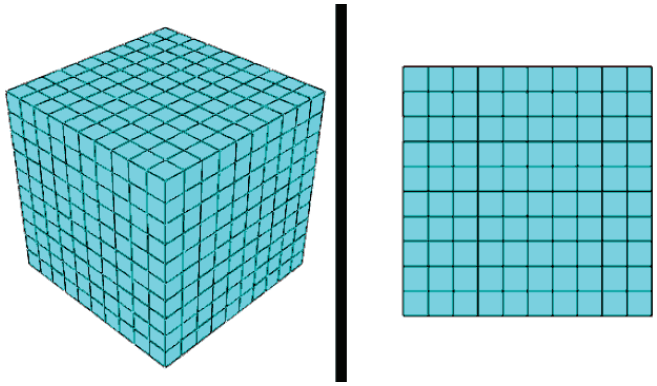
2026 FD01

Surface Tension Loads in Abaqus/Standard

You can now specify surface tension loads on element-based surfaces in Abaqus/Standard.

Benefits: This feature expands the capabilities of Abaqus/Standard.
In earlier releases, surface tension loads on element-based surfaces were available only in Abaqus/Explicit.

The animation below shows a cube being transformed into a sphere under surface tension loads in Abaqus/Standard. The cube is modeled as an elastic plastic material with a very small yield stress using C3D8H elements in a static analysis. A surface tension load is applied on all facets of the elements forming the exterior face.



For more information, see **DSLOAD* and *Surface Tension Loads*.

Long-Term Stress Initial Conditions

You can now specify initial stresses for viscoelastic materials in a fully relaxed state.

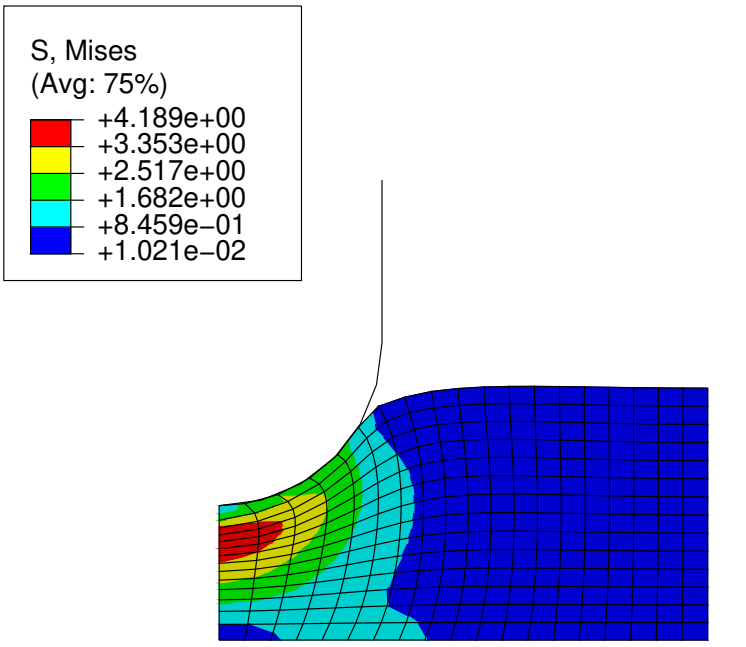
Benefits: This feature expands the capabilities of Abaqus/Standard and Abaqus/Explicit.

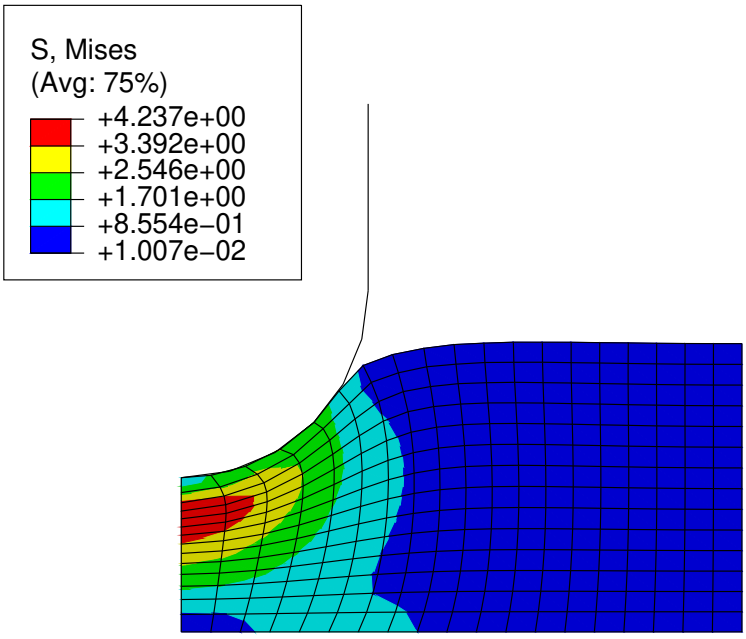
By default, when you specify initial stresses for a viscoelastic material, Abaqus assumes that they are instantaneous (that is, applied very fast). Now, you can specify initial stresses in the case where load is applied for a long enough time for the material to reach a fully relaxed state. An example of such a situation is a part that is preloaded during installation and then left in this state for a long time. In this case, if the stresses at the end of this period are known and are used to specify initial stresses, you should use the new long-term initial stress capability.

To illustrate this capability, we consider a specimen that is indented by a hemispherical punch. The specimen is modeled using a hyperelastic material with finite-strain linear viscoelasticity. Two Abaqus/Standard analyses are performed as follows:

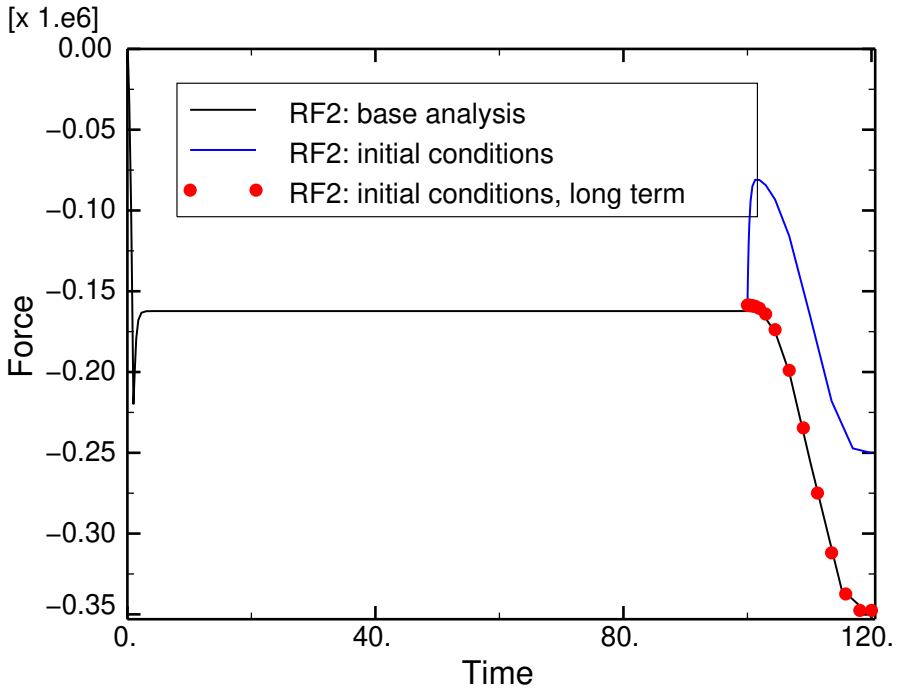
- The first analysis consists of two steps. In the first step, the punch moves down and then stays in this position long enough to reach the fully relaxed state. In the second step, the punch is moved down again.
- The initial configuration of the second analysis is created using the deformed configuration at the end of the first step of the previous analysis. The long-term initial stresses are specified, which correspond to the end of the first step. Finally, the punch is moved down the same way as in the second step of the first analysis.

The contour plots below show the stresses at the end of the second step of the first analysis (top) and at the end of the second analysis (bottom). It is expected that these results will match, and they show a good agreement.


S, Mises
(Avg: 75%)
+4.189e+00
+3.353e+00
+2.517e+00
+1.682e+00
+8.459e−01
+1.021e−02


S, Mises
(Avg: 75%)
+4.237e+00
+3.392e+00
+2.546e+00
+1.701e+00
+8.554e−01
+1.007e−02

Similarly, the reaction forces at the punch reference node correlate well, as shown below. However, the reaction force for the case when instantaneous initial stresses are prescribed instead of long-term stresses shows substantial discrepancies.



For more information, see *Defining Initial Stresses*.

Initial Stress for Materials with Mullins Effect

You can now specify initial stresses for materials with any of the Mullins effect models available in Abaqus.

Benefits: This feature expands the capabilities of Abaqus/Standard and Abaqus/Explicit.

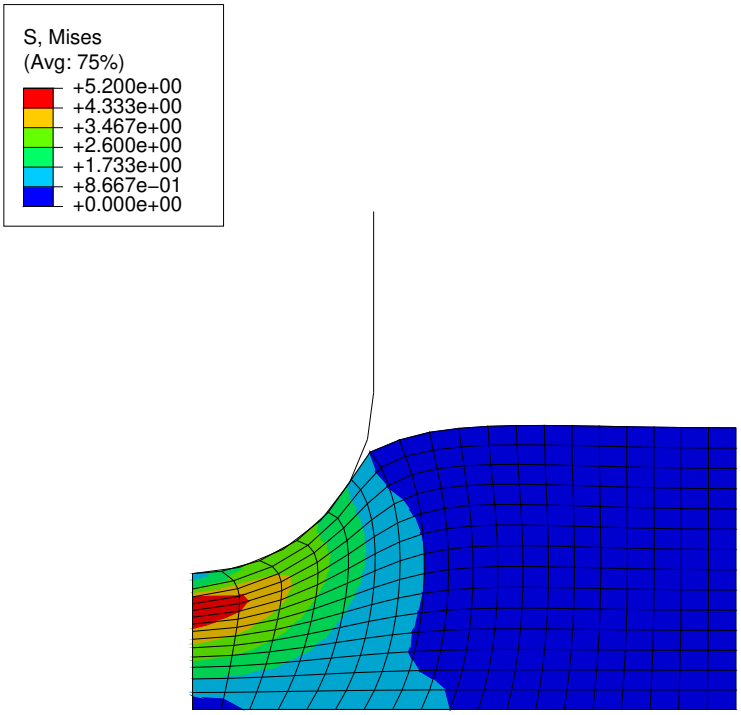
In earlier releases, initial stresses were supported only for materials using the Ogden-Roxburgh or user-defined Mullins effect models. Now, you can specify initial stresses for materials using any of the Mullins effect models available in Abaqus. In addition, you can now use initial stresses in conjunction with the initial maximum energy density in the Mullins effect model to determine if the stress lies on the loading or the unloading path. If the initial maximum energy density is larger than that computed using the initial stresses, the stress lies on the unloading path; otherwise, the stress lies on the loading path.

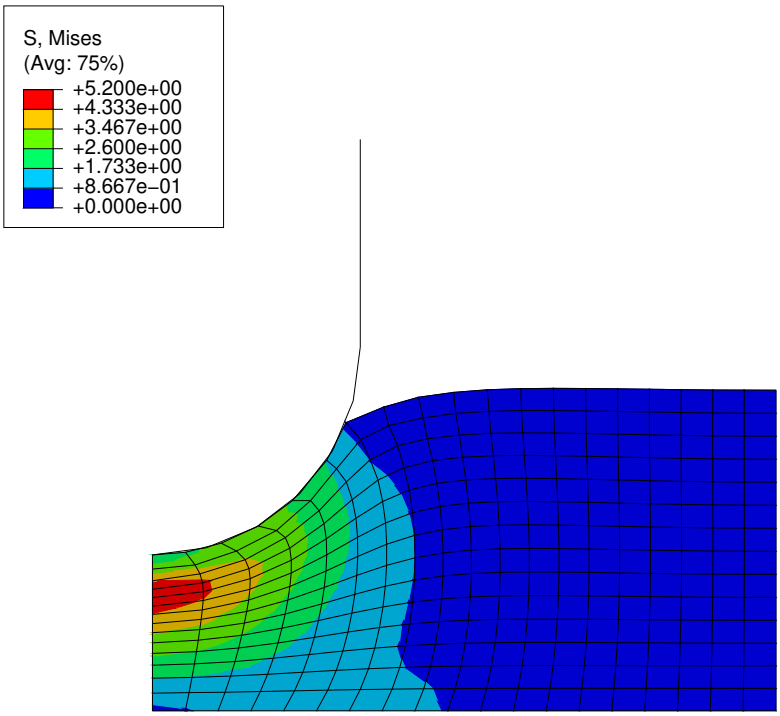
In the example below, a specimen modeled using a hyperelastic material with the Ogden-Roxburgh Mullins effect model is subjected to a nonmonotonic load by a hemispherical punch. Two analyses are performed as follows:

- First, a two-step Abaqus/Standard analysis is performed. In the first step, the punch is moved down by 120 mm and then moved up by 30 mm. In the second step, the punch is moved down again by 60 mm.
- The initial configuration of the second, Abaqus/Explicit, analysis corresponds to the deformed configuration of the Abaqus/Standard analysis at the end of the first step. The initial stresses and initial maximum energy

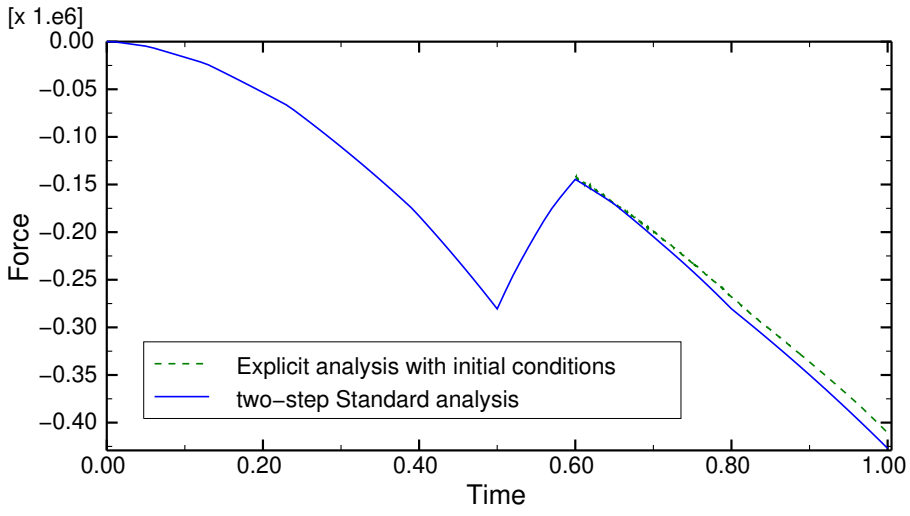
density of the Mullins model are prescribed from the end of the first step of the Abaqus/Standard analysis. The second analysis consists of one step in which the punch is moved down by 60 mm.

The contour plots below show the stress results at the end of the first analysis (top) and at the end of the second analysis (bottom).





The reaction forces in the punch reference node are plotted below.



Both the stress results and the reaction forces show good agreement as expected.

For more information, see *Defining Initial Stresses*.

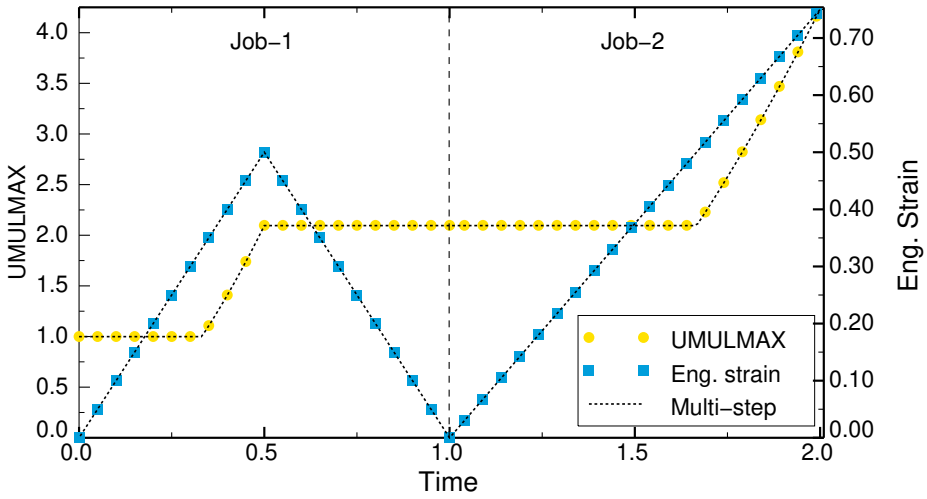
Initial Maximum Energy Density in Mullins Effect Models

You can now define the initial value of the maximum energy density used in Mullins effect models.

Benefits: This feature expands the capabilities of Abaqus/Standard and Abaqus/Explicit.

To specify the initial value of the maximum energy density used in Mullins effect models, use the new **INITIAL CONDITIONS*, TYPE=MULLINS option. If required, you can use the **EXTERNAL FIELD* option in conjunction with this option.

The plot below shows the evolution of UMULMAX across two separate simulations, Job-1 and Job-2, of a unit cube subjected to uniaxial tension.



The value of UMULMAX is set to 1.0 at the start of Job-1 without the use of **EXTERNAL FIELD* and reaches a value of 2.09 upon deformation. This new value is set as the initial condition for Job-2 using **EXTERNAL FIELD*. These results match the corresponding results for the equivalent single multistep simulation that are included for reference.

For more information, see *Defining Initial Maximum Energy Density in Mullins Effect Models*.

2026 GA

Contact-Dependent Thermal Radiation Surface Loads in Abaqus/Explicit

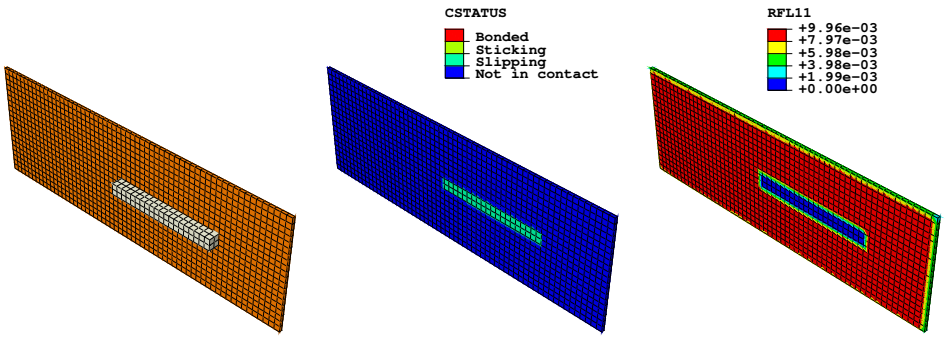
You can now have surface-based thermal radiation loads automatically suspended where contact is active in Abaqus/Explicit. In earlier releases, this functionality was available only in Abaqus/Standard.

Benefits: You can model surface-based thermal loads more accurately by accounting for shielding effects of contact on these loads.

In earlier releases, thermal radiation loads on individual surfaces associated with surface-based radiation to ambient temperature were active regardless of the contact status in Abaqus/Explicit. Now, you can model shielding of surface-based radiation to ambient temperature due to active contact.

To use the active contact status information to turn off heat exchange from surface radiation loads, you can explicitly define surface-based radiation to depend on the contact status. Alternatively, if thermal shielding due to contact is not desired, you can turn off contact status dependence for surface-based radiation.

The animation below shows two blocks in contact and maintained at a constant temperature with specified temperature boundary conditions. The blocks exchange heat with their surroundings with prescribed surface-based radiation on the exterior surfaces in a coupled temperature-displacement analysis. Active contact regions shield the thermal radiation by default and evolve as the top block moves. The shielded regions with zero reactive heat flux RFL11 also evolve accordingly.



For more information, see **SRADIATE* and *Thermal Loads*.

2025 FD03

User-Defined Electrical Current Density Values

In Abaqus/Standard, you can now define nonuniform electrical current density values using the new user subroutine `DECURRENT`.

Benefits: You can account for electrical current and its effects in a variety of scientific, industrial, and other applications.
You can define the electrical current density value as a function of position, time, element number, field variables, property and parameter tables in table collections, etc. for coupled thermal-electrical, coupled temperature-displacement electrical, coupled thermal-electrochemical, and coupled temperature-displacement electrochemical analyses.

For more information, see **DECURRENT*, **DSECURRENT*, and *Electromagnetic Loads*.

Surface Tension Loads in Abaqus/Explicit

You can now specify surface tension loads on element-based surfaces in Abaqus/Explicit.

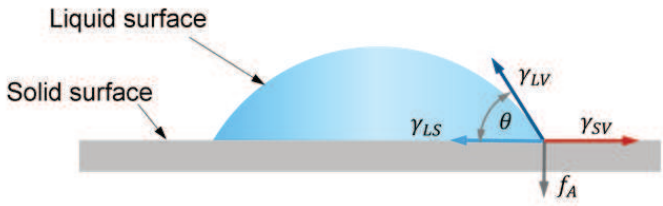
Benefits: You can account for surface tension and its effects in a variety of scientific, industrial, and medical applications.

Surface tension is a physical property of liquids that arises from cohesive forces between molecules at an interface. It behaves like an invisible, elastic membrane on the liquid surface and plays a critical role in the analysis of liquid interfaces and deformable structures in various scientific, industrial, and medical applications. In computational modeling, accurately representing surface tension is essential for capturing phenomena such as

droplet formation, capillary effects, and fluid-structure interaction for processes such as solder reflow or glass manufacturing.

Surface tension loads in Abaqus/Explicit enable you to simulate these effects in soft or fluid-like materials. You can specify the contact angle, which is defined as the equilibrium angle between the liquid-solid and liquid-vapor interfaces, to represent surface wettability characteristics that directly influence interface shape and movement.

The image below shows the equilibrium configuration of a fluid in contact with a solid. The contact angle, θ, is the equilibrium angle formed between the liquid-solid interface and the liquid-vapor interface due to the combined effect of the liquid-vapor surface tension, γ_{LV}, the solid-vapor surface tension, γ_{SV}, and the liquid-solid interfacial tension, γ_{LS}. The contact angle is a measure of surface's wettability, describing the tendency of the liquid to spread or bead up.



The animation below shows the contact of a cubic solder droplet with a pad under surface tension loads. A surface tension load is applied on each exterior face. The melted state of the solder material is modeled using a linear equation of state (EOS) material of type USUP. A no-separation, free-slip condition is enforced for the contact between the solder and pad. The contact angle θ is defined as 5 degrees.

For more information, see **DSLOAD* and *Surface Tension Loads*.

2025 FD02

Contact-Dependent Thermal Radiation Surface Loads

You can now deactivate surface-based thermal radiation loads on portions of surfaces in active contact.

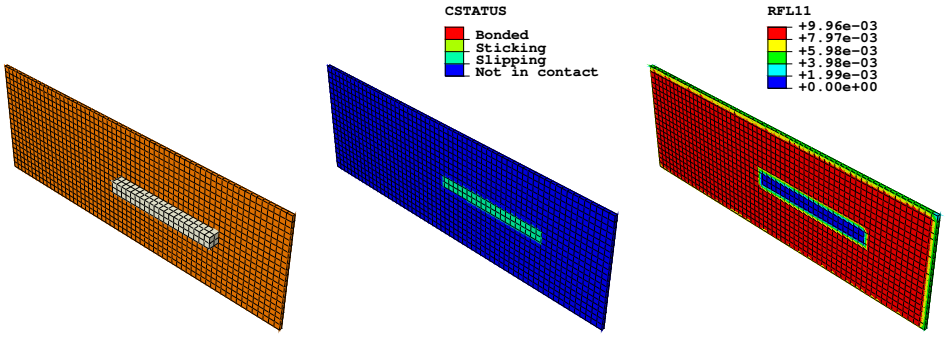
Benefits: You can model surface-based thermal loads more accurately by accounting for the effect of contact on these loads.

In earlier releases, thermal radiation loads on individual surfaces assumed that the surrounding environment was nonreflective. In this release, you can model thermal shielding on a surface due to active contact with a fully reflective surface. By default, active contact regions turn off surface thermal radiation loads altogether.

To use the active contact status information to turn off heat exchange from surface radiation loads, set the new TRANSITION parameter on **SRADIATE* equal to TOUCHING. Alternatively, if thermal shielding is not

effective, you can retain radiation heat exchange with the surroundings by setting TRANSITION=NONE to ignore active contact.

The animation below shows two blocks in contact and maintained at a constant temperature with specified temperature boundary conditions. The blocks exchange heat with surroundings with prescribed surface-based radiation on the exterior surfaces in a coupled temperature-displacement analysis. Active contact regions shield thermal radiation by default and evolve as the top block moves. The shielded regions with zero reactive heat flux RFL11 also evolve accordingly.



For more information, see **SRADIATE* and *Thermal Loads*.

Contact-Dependent Surface-Based Thermal Convection Loads in Abaqus/Explicit

You can now disable surface-based thermal convection loads automatically on portions of surfaces in active contact in Abaqus/Explicit. This functionality is previously available in Abaqus/Standard.

Benefits: You can more accurately model shielding of portions of a surface in active contact from specified thermal convection loads. In addition, it is easier to select surfaces for surface-based thermal convection loads.

Abaqus internally determines active contact regions and automatically turns off thermal convection loads on such regions. Surfaces with prescribed thermal film conditions model convective heat exchange with the surrounding gas or fluid to which they are directly exposed. When portions of these surfaces are in active contact with other surfaces, convective heat exchange with the surrounding gas or fluid is negligible due to shielding from active contact.

For more information, see **SFILM* and *Thermal Loads*.

2025 FD01

Contact-Dependent Surface-Based Thermal Convection Loads

You can now disable surface-based thermal convection loads automatically on portions of surfaces in active contact.

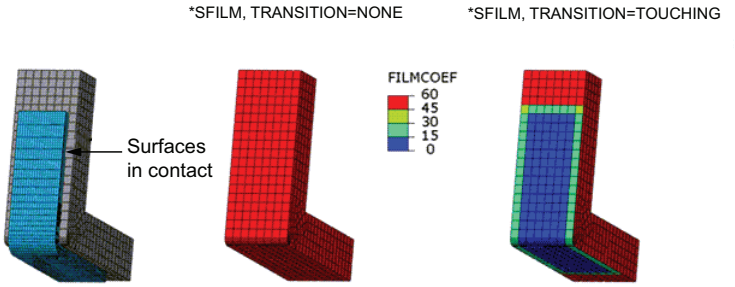
Benefits: You can more accurately model shielding of portions of a surface in active contact from specified thermal convection loads. In addition, it is easier to select surfaces for surface-based thermal convection loads.

Abaqus internally determines active contact regions and automatically turns off thermal convection loads on such regions. Surfaces with prescribed thermal film conditions model convective heat exchange with the

surrounding gas or fluid to which they are directly exposed. When portions of these surfaces are in active contact with other surfaces, convective heat exchange with the surrounding gas or fluid is negligible due to shielding from active contact.

You can set the new TRANSITION parameter equal to TOUCHING to use the active contact status information to turn off convective heat exchange. This value is the default. Alternatively, you can set TRANSITION=NO to ignore the dependence on the current contact status.

The image below shows two angle brackets in contact. Surface-based convective thermal loads are prescribed on the exterior surfaces of the brackets by specifying a uniform film coefficient value of 60.



The contour plot on the right shows a film coefficient of zero on the larger bracket in regions with active contact in order to shield convective heat exchange with the surroundings. In earlier releases, you had to select the surfaces carefully to exclude active contact regions, which is cumbersome and impractical especially when active contact regions evolve.

For more information, see **SFILM* and *Thermal Loads*.

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 Prescribed Conditions Guide*.

2024 FD01

Modified Interface to Define Time Variation of Ambient Temperature for Thermal Radiation

For prescribed thermal radiation boundary conditions, the new AMBIENT AMPLITUDE replaces the AMPLITUDE parameter available in earlier versions.

Benefits: The new parameter name better describes the capability, which enhances the clarity of the user interface.

The new AMBIENT AMPLITUDE parameter behaves identically to the AMPLITUDE parameter available in earlier releases. Abaqus will continue to support the AMPLITUDE parameter for some time; however, you are encouraged to replace it with the new parameter in all your simulations.

For more information, see **RADIATE*, **CRADIATE*, **SRADIATE*, and *Thermal Loads*.

Time Variation of Radiation Flux

For prescribed thermal radiation boundary conditions, you can use the new RADIATION AMPLITUDE parameter to define the time variation of the prescribed radiation flux directly through an amplitude definition.

Benefits: This capability provides additional modeling flexibility in some situations.

For more information, see **RADIATE*, **CRADIATE*, **SRADIATE*, and *Thermal Loads*.

Modified Interface to Define Time Variation of Sink Temperature for Convection Boundary Conditions

For prescribed convective (film) boundary conditions in a thermal analysis, the new SINK AMPLITUDE parameter replaces the AMPLITUDE parameter available in earlier versions.

Benefits: The new parameter name better describes the capability, which enhances the clarity of the user interface. The new SINK AMPLITUDE parameter behaves identically to the AMPLITUDE parameter available in earlier releases. Abaqus will continue to support the AMPLITUDE parameter for some time; however, you are encouraged to replace it with the new parameter in all your simulations.

For more information, see **FILM*, **CFILM*, **SFILM*, and *Thermal Loads*.

Algorithmic Improvements to Convection and Radiation Boundary Flux Calculations

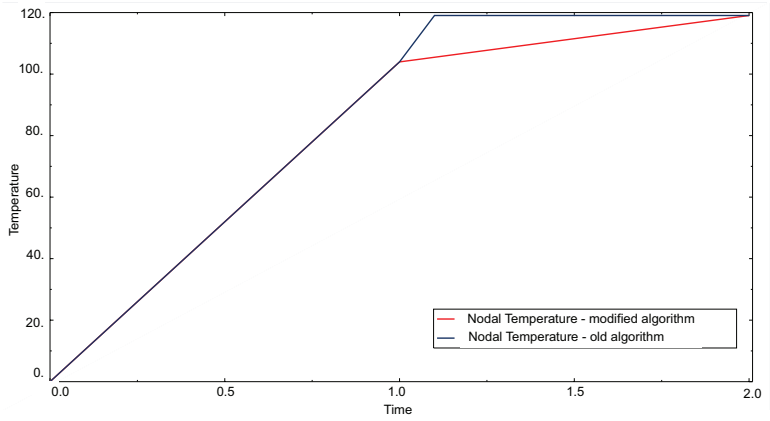
The flux calculations associated with prescribed convection and/or radiation boundary conditions in Abaqus/Standard now use an improved algorithm.

Benefits: The improved algorithms ensure continuity of convective/radiative fluxes across step boundaries, resulting in smooth temperature variations, and lead to better overall convergence characteristics for certain types of multi-step steady-state thermal analyses.

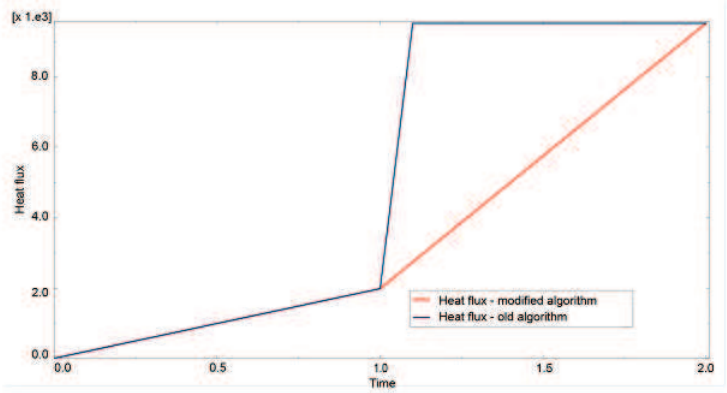
By default, in the first step of a steady-state thermal analysis earlier versions of Abaqus/Standard assumed that the sink temperature was ramped up from zero to its specified value over the step time period. In the presence of prescribed nonzero initial temperatures, this assumption led to discontinuities in the convective/radiative fluxes, as well as the temperature fields. In addition, in a multi-step analysis, the default amplitude behavior associated with film/radiation thermal loads could result in nonsmooth temperature and heat fluxes across step boundaries. Such discontinuities could also lead to convergence difficulties in nonlinear problems. The new, improved algorithm removes the default modulation of the sink temperature by the step amplitude, ensuring continuity of fluxes, smooth variations of temperature, and better overall convergence. For more information, see the Knowledge Base item QA00000307343 in the Dassault Systèmes Knowledge Base.

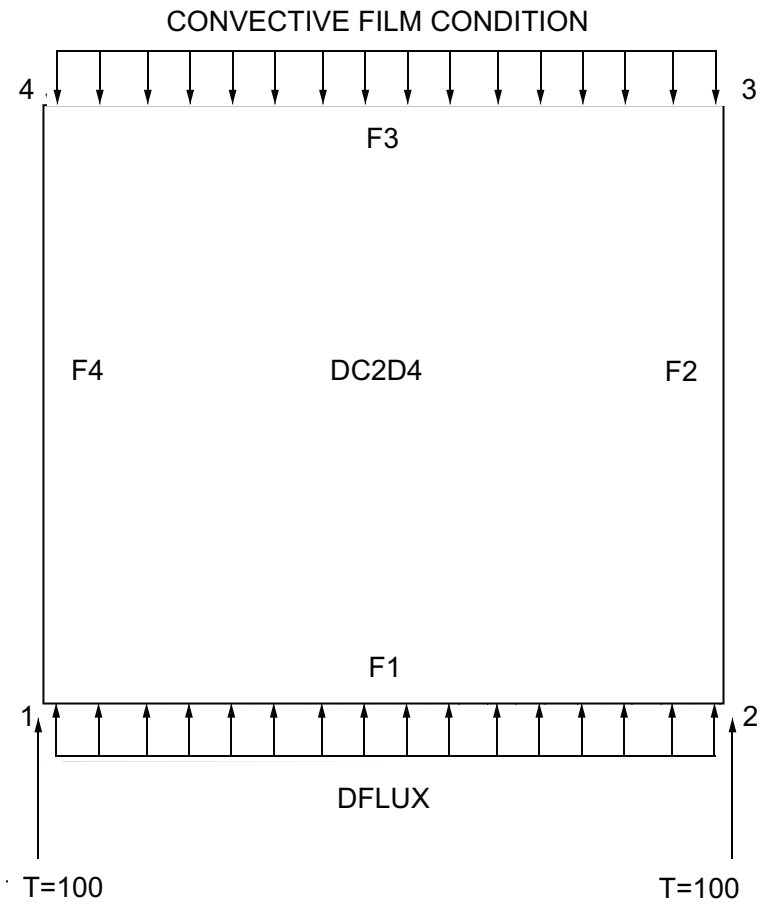
The figure below shows a one-element test with prescribed temperatures at the bottom nodes, a prescribed heat flux along the bottom surface, and a prescribed convective (film) boundary condition on the top surface. There are two steady-state heat transfer steps in the analysis, each with a time period of 1.0 (for a total time period of 2.0).

The plot of temperature versus time below contrasts the temperatures across step transitions using the old and the improved (modified) algorithms.



The plot of convective heat flux versus time below contrasts the convective heat fluxes across step transitions using the old and the improved (modified) algorithms.





In the old formulation, the convective heat flux experienced a discontinuity at the transition between the two steps. The improved formulation leads to a smoother behavior of the convective heat flux over the second step.

For more information, see **FILM*, **CFILM*, **SFILM*, **RADIATE*, **CRADIATE*, **SRADIATE*, and *Thermal Loads*.

About Prescribed Conditions

Overview

Prescribed conditions allow you to model nonzero initial conditions, boundary conditions, loads, and predefined fields.

References:

- *About User Subroutines and Utilities*

Amplitude Variations

Complex time- or frequency-dependent boundary conditions, loads, and predefined fields can be specified by referring to an amplitude curve in the prescribed condition definition. Amplitude curves are explained in *Amplitude Curves*.

In Abaqus/Standard if no amplitude is referenced from the boundary condition, loading, or predefined field definition, the total magnitude can be applied instantaneously at the start of the step and remain constant throughout the step (a “step” variation) or it can vary linearly over the step from the value at the end of the previous step (or from zero at the start of the analysis) to the magnitude given (a “ramp” variation). You choose the type of variation when you define the step; the default variation depends on the procedure chosen, as shown in *Defining an Analysis*.

In Abaqus/Standard the variation of many prescribed conditions can be defined in user subroutines. In this case the magnitude of the variable can vary in any way with position and time. The magnitude variation for prescribing and removing conditions must be specified in the subroutine (see *About User Subroutines and Utilities*).

In Abaqus/Explicit if no amplitude is referenced from the boundary condition or loading definition, the total value will be applied instantaneously at the start of the step and will remain constant throughout the step (a “step” variation), although Abaqus/Explicit does not admit jumps in displacement (see *Boundary Conditions*). If no amplitude is referenced from a predefined field definition, the total magnitude will vary linearly over the step from the value at the end of the previous step (or from zero at the start of the analysis) to the magnitude given (a “ramp” variation).

When boundary conditions are removed (see *Boundary Conditions*), the boundary condition (displacement or rotation constraint in stress/displacement analysis) is converted to an applied conjugate flux (force or moment in stress/displacement analysis) at the beginning of the step. This flux magnitude is set to zero with a “step” or “ramp” variation depending on the procedure chosen, as discussed in *Defining an Analysis*. Similarly, when loads and predefined fields are removed, the load is set to zero and the predefined field is set to its initial value.

Applying Boundary Conditions and Loads in a Local Coordinate System

You can define a local coordinate system at a node as described in *Transformed Coordinate Systems*. Then, all input data for concentrated force and moment loading and for displacement and rotation boundary conditions are given in the local system.

Loads and Predefined Fields Available for Various Procedures

Table 1: Available loads and predefined fields.

Loads and predefined fields	Procedures
Added mass (concentrated and distributed)	Abaqus/Aqua eigenfrequency extraction analysis (*Natural Frequency Extraction*)
Base motion	Procedures based on eigenmodes:
	Transient Modal Dynamic Analysis
	Mode-Based Steady-State Dynamic Analysis
	Response Spectrum Analysis
	Random Response Analysis
Boundary condition with a nonzero prescribed boundary	All procedures except those based on eigenmodes
Connector motion Connector load	All relevant procedures except modal extraction, buckling, those based on eigenmodes, and direct steady-state dynamics
Cross-correlation property	*Random Response Analysis*
Current density (concentrated and distributed)	*Coupled Thermal-Electrical Analysis*
	Fully Coupled Thermal-Electrical-Structural Analysis
Current density vector	*Eddy Current Analysis*
Electric charge (concentrated and distributed)	*Piezoelectric Analysis*
Equivalent pressure stress	*Mass Diffusion Analysis*
Film coefficient and associated sink temperature	All procedures involving temperature degrees of freedom
Fluid flux	Analysis involving hydrostatic fluid elements
Fluid mass flow rate	Analysis involving convective heat transfer elements
Flux (concentrated and distributed)	All procedures involving temperature degrees of freedom *Mass Diffusion Analysis*
Force and moment (concentrated and distributed)	All procedures with displacement degrees of freedom except response spectrum
Incident wave loading	Direct-integration dynamic analysis (*Implicit Dynamic Analysis Using Direct Integration*) involving solid and/or fluid elements undergoing shock loading
Pore fluid pressure as a predefined field	*Static Stress Analysis*
	Explicit Dynamic Analysis
Predefined field variable	All procedures except those based on eigenmodes
Seepage coefficient and associated sink pore pressure Distributed seepage flow	*Coupled Pore Fluid Diffusion and Stress Analysis*
Substructure load	All procedures involving the use of substructures
Temperature as a predefined field	All procedures except adiabatic analysis, mode-based procedures, and procedures involving temperature degrees of freedom

With the exception of concentrated added mass and distributed added mass, no loads can be applied in eigenfrequency extraction analysis.

Amplitude Curves

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Prescribed Conditions*
- **AMPLITUDE*
- *The Amplitude toolset*

Overview

You can use amplitude curves to define arbitrary time (or frequency) variations of load, displacement, and other prescribed variables to be given throughout a step (using step time) or throughout the analysis (using total time).

An amplitude curve:

- can be defined as a mathematical function (such as a sinusoidal variation), as a series of values at points in time (such as a digitized acceleration-time record from an earthquake), as a user-customized definition via user subroutines, or, in Abaqus/Standard, as values calculated based on a solution-dependent variable (such as the maximum creep strain rate in a superplastic forming problem); and
- can be referred to by name by any number of boundary conditions, loads, and predefined fields.

Amplitude Curves

By default, the values of loads, boundary conditions, and predefined fields either change linearly with time throughout the step (ramp function) or they are applied immediately and remain constant throughout the step (step function)—see *Defining an Analysis*. Many problems require a more elaborate definition, however. For example, different amplitude curves can be used to specify time variations for different loadings. One common example is the combination of thermal and mechanical load transients: usually the temperatures and mechanical loads have different time variations during the step. Different amplitude curves can be used to specify each of these time variations.

Other examples include dynamic analysis under earthquake loading, where an amplitude curve can be used to specify the variation of acceleration with time, and underwater shock analysis, where an amplitude curve is used to specify the incident pressure profile.

Amplitudes are defined as model data (that is, they are not step dependent). Each amplitude curve must be named; this name is then referred to from the load, boundary condition, or predefined field definition (see *About Prescribed Conditions*).

Input File Usage: **AMPLITUDE*, NAME=*name*

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **Name:** *name*

Defining the Time Period

Each amplitude curve is a function of time or frequency. Amplitudes defined as functions of frequency are used in *Direct-Solution Steady-State Dynamic AnalysisMode-Based Steady-State Dynamic Analysis* and *Eddy Current Analysis*.

Amplitudes defined as functions of time can be given in terms of *step time* (default) or in terms of *total time*. These time measures are defined in *Conventions*.

Input File Usage: Use one of the following options:

**AMPLITUDE*, NAME=*name*, TIME=STEP TIME (default)
**AMPLITUDE*, NAME=*name*, TIME=TOTAL TIME

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: any type: **Time span: Step time** or **Total time**

Continuation of an Amplitude Reference in Subsequent Steps

If a boundary condition, load, or predefined field refers to an amplitude curve and the prescribed condition is not redefined in subsequent steps, the following rules apply:

- If the associated amplitude is given in terms of total time, the prescribed condition continues to follow the amplitude definition.
- If the amplitude is given in terms of step time, the following rules apply:
  - Boundary conditions and predefined fields remain constant at the magnitude associated with the end of the previous step.
  - Loads remain constant at the magnitude associated with the end of the previous step unless the amplitude is specified in user subroutine *UAMP* (Abaqus/Standard) or *VUAMP* (Abaqus/Explicit), in which case the loads are removed immediately.
- If no associated amplitude is given, the prescribed condition remains constant at the magnitude associated with the end of the previous step.

Specifying Relative or Absolute Data

You can choose between specifying relative or absolute magnitudes for an amplitude curve.

Relative Data

By default, you give the amplitude magnitude as a multiple (fraction) of the reference magnitude given in the prescribed condition definition. This method is especially useful when the same variation applies to different load types.

Input File Usage: **AMPLITUDE*, NAME=*name*, VALUE=RELATIVE

Abaqus/CAE Usage: Amplitude magnitudes are always relative in Abaqus/CAE.

Absolute Data

Alternatively, you can give absolute magnitudes directly. When this method is used, the values given in the prescribed condition definitions are ignored.

In general, you should not use absolute amplitude values to define temperatures or predefined field variables for nodes attached to beam or shell elements as values at the reference surface together with the gradient or gradients across the section (default cross-section definition; see *Using a Beam Section Integrated during the Analysis to Define the Section Behavior* and *Using a Shell Section Integrated during the Analysis to Define the Section Behavior*). Because the values given in temperature fields and predefined fields are ignored, the absolute amplitude value is used to define both the temperature and the gradient and field and gradient, respectively.

Input File Usage: **AMPLITUDE*, NAME=*name*, VALUE=ABSOLUTE

Abaqus/CAE Usage: Absolute amplitude magnitudes are not supported in Abaqus/CAE.

Defining the Amplitude Data

The variation of an amplitude with time can be specified in several ways. The variation of an amplitude with frequency can be given only in tabular or equally spaced form.

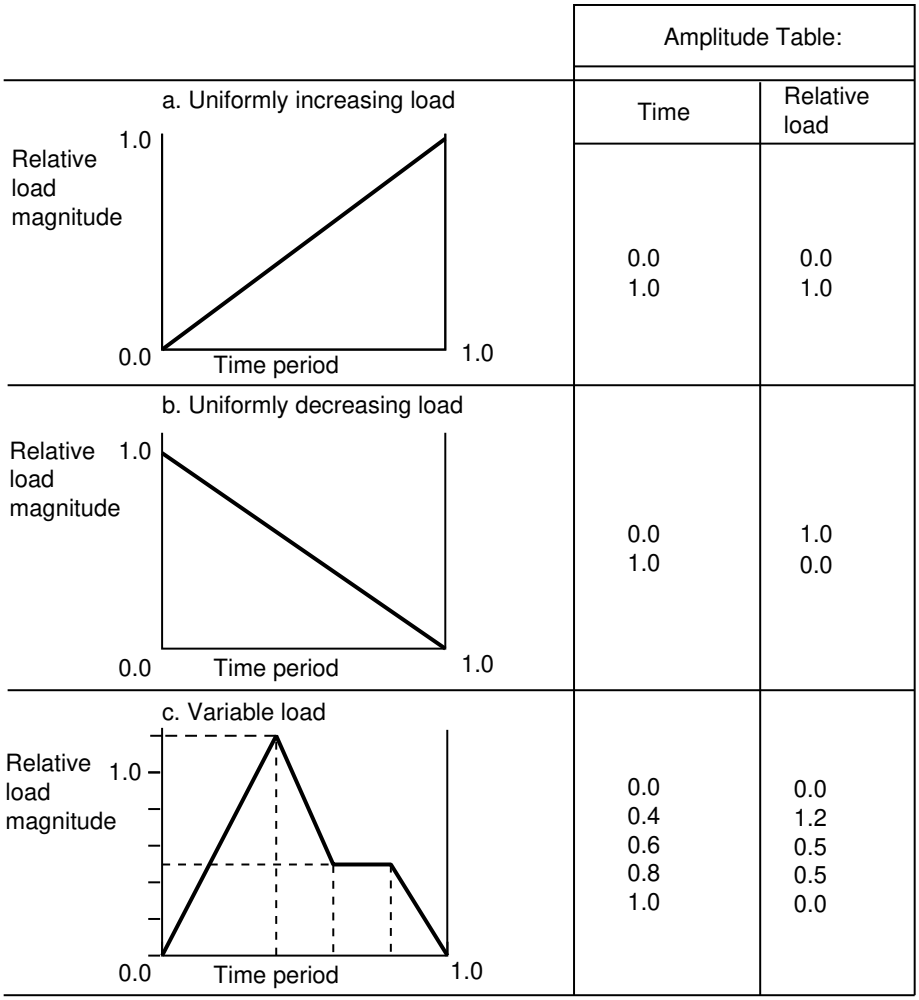
Defining Tabular Data

Choose the tabular definition method (default) to define the amplitude curve as a table of values at convenient points on the time scale. Abaqus interpolates linearly between these values, as needed. By default in Abaqus/Standard, if the time derivatives of the function must be computed, some smoothing is applied at the time points where the time derivatives are discontinuous. In contrast, in Abaqus/Explicit no default smoothing is applied (other than the inherent smoothing associated with a finite time increment). You can modify the default smoothing values (smoothing is discussed in more detail below, under the heading “Using an amplitude definition with boundary conditions”); alternatively, a smooth step amplitude curve can be defined (see “Defining smooth step data” below).

If the amplitude varies rapidly—as with the ground acceleration in an earthquake, for example—you must ensure that the time increment used in the analysis is small enough to pick up the amplitude variation accurately since Abaqus samples the amplitude definition only at the times corresponding to the increments being used.

If the analysis time in a step is less than the earliest time for which data exist in the table, Abaqus applies the earliest value in the table for all step times less than the earliest tabulated time. Similarly, if the analysis continues for step times past the last time for which data are defined in the table, the last value in the table is applied for all subsequent time.

Several examples of tabular input are shown in *Figure 1*.



Amplitude Table:	
Time	Relative load
0.0 1.0	0.0 1.0
0.0 1.0	1.0 0.0
0.0 0.4 0.6 0.8 1.0	0.0 1.2 0.5 0.5 0.0

Figure 1: Tabular amplitude definition examples.

Input File Usage: **AMPLITUDE*, NAME=*name*, DEFINITION=TABULAR

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **Tabular**

Defining Equally Spaced Data

Choose the equally spaced definition method to give a list of amplitude values at fixed time intervals beginning at a specified value of time. Abaqus interpolates linearly between each time interval. You must specify the fixed time (or frequency) interval at which the amplitude data is given, Δt. You can also specify the time (or lowest frequency) at which the first amplitude is given, t_0; the default is t_0=0.0.

If the analysis time in a step is less than the earliest time for which data exist in the table, Abaqus applies the earliest value in the table for all step times less than the earliest tabulated time. Similarly, if the analysis continues for step times past the last time for which data are defined in the table, the last value in the table is applied for all subsequent time.

Input File Usage: **AMPLITUDE*, NAME=*name*, DEFINITION=EQUALLY SPACED,
FIXED INTERVAL=Δt, BEGIN=t_0

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **Equally spaced**: **Fixed interval:** Δt

The time (or lowest frequency) at which the first amplitude is given, t_0, is indicated in the first table cell.

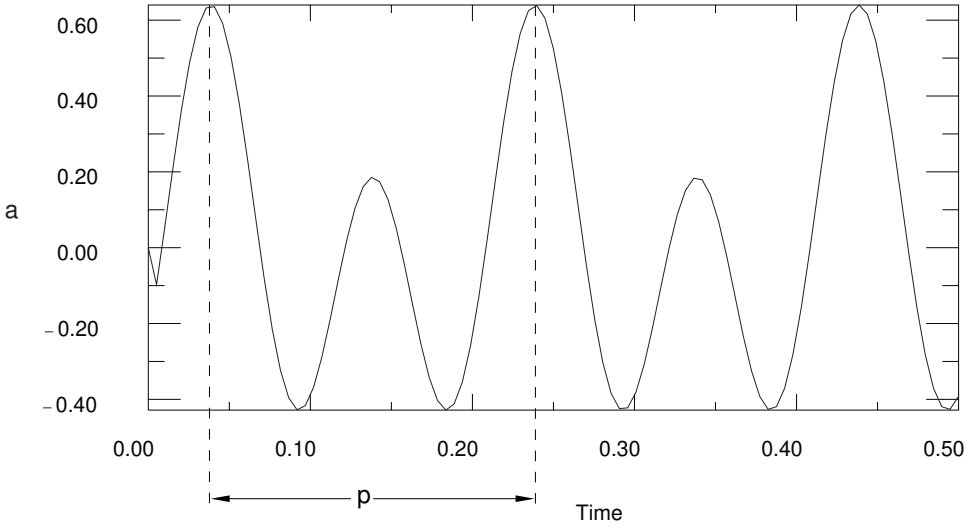
Defining Periodic Data

Choose the periodic definition method to define the amplitude, a, as a Fourier series:

$$a = A_0 + \sum_{n=1}^{N} [A_n \cos n\omega (t - t_0) + B_n \sin n\omega (t - t_0)] \text{ for } t \geq t_0,$$

$$a = A_0 \quad \text{for } t < t_0,$$

where t_0, N, ω, A_0, A_n, and B_n, $n = 1, 2 \ldots N$, are user-defined constants. An example of this form of input is shown in *Figure 2*.



p = 0.2s

$a = A_0 + \sum_{n=1}^{N} [A_n \cos n\omega(t-t_0) + B_n \sin n\omega(t-t_0)]$ for $t \geq t_0$

$a = A_0$ for $t < t_0$

with

N = 2, ω = 31.416 rad/s, $t_0 = -0.1614$ s

$A_0 = 0$, $A_1 = 0.227$, $B_1 = 0.0$, $A_2 = 0.413$, $B_2 = 0.0$

Figure 2: Periodic amplitude definition example.

Input File Usage: **AMPLITUDE*, NAME=*name*, DEFINITION=PERIODIC

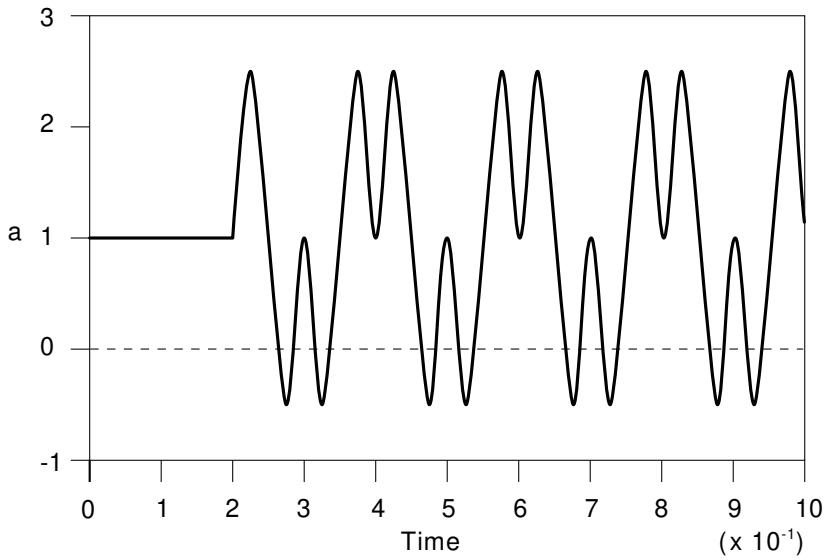
Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **Periodic**

Defining Modulated Data

Choose the modulated definition method to define the amplitude, a, as

$$a = A_0 + A \sin \omega_1 (t - t_0) \sin \omega_2 (t - t_0) \quad \text{for } t > t_0,$$
$$a = A_0 \quad \text{for } t \le t_0,$$

where A_0, A, t_0, ω_1, and ω_2 are user-defined constants. An example of this form of input is shown in *Figure 3*.



$$a = A_0 + A \sin \omega_1 (t - t_0) \sin \omega_2 (t - t_0) \quad \text{for } t > t_0$$

$$a = A_0 \quad \text{for } t \le t_0$$

with

$$A_0 = 1.0, \quad A = 2.0, \quad \omega_1 = 10\pi, \quad \omega_2 = 20\pi, \quad t_0 = .2$$

Figure 3: Modulated amplitude definition example.

Input File Usage: **AMPLITUDE*, NAME=*name*, DEFINITION=MODULATED

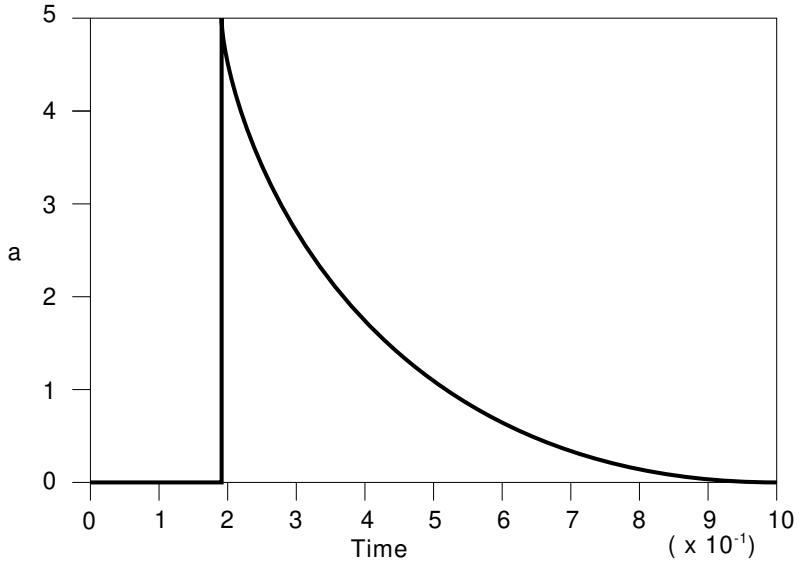
Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **Modulated**

Defining Exponential Decay

Choose the exponential decay definition method to define the amplitude, a, as

$$a = A_0 + A \exp\left(-(t - t_0)/t_d\right) \quad \text{for } t \ge t_0,$$
$$a = A_0 \quad \text{for } t < t_0,$$

where A_0, A, t_0, and t_d are user-defined constants. An example of this form of input is shown in *Figure 4*.



$a = A_0 + A \exp[-(t-t_0)/t_d]$ for $t \geq t_0$

$a = A_0$ for $t < t_0$

with

$A_0 = 0.0$, $A = 5.0$, $t_0 = 0.2$, $t_d = 0.2$

Figure 4: Exponential decay amplitude definition example.

Input File Usage: **AMPLITUDE*, NAME=*name*, DEFINITION=DECAY

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **Decay**

Defining Smooth Step Data

Abaqus/Standard and Abaqus/Explicit can calculate amplitudes based on smooth step data. Choose the smooth step definition method to define the amplitude, *a*, between two consecutive data points (t_i, A_i) and (t_{i+1}, A_{i+1}) as

$$a = A_i + (A_{i+1} - A_i)\,\xi^3\,(10 - 15\xi + 6\xi^2) \quad \text{for } t_i \leq t \leq t_{i+1},$$

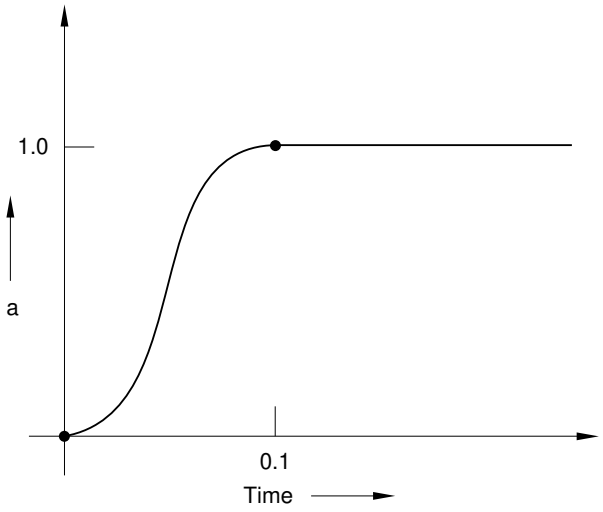
where $\xi = (t - t_i) / (t_{i+1} - t_i)$. The above function is such that $a = A_i$ at t_i, $a = A_{i+1}$ at t_{i+1}, and the first and second derivatives of *a* are zero at t_i and t_{i+1}. This definition is intended to ramp up or down smoothly from one amplitude value to another.

The amplitude, *a*, is defined such that

$$a = A_0 \quad \text{for } t \leq t_0,$$
$$a = A_f \quad \text{for } t \geq t_f,$$

where (t_0, A_0) and (t_f, A_f) are the first and last data points, respectively.

Examples of this form of input are shown in *Figure 5* and *Figure 6*. This definition cannot be used to interpolate smoothly between a set of data points; that is, this definition cannot be used to do curve fitting.



$t_0 = 0.0 \qquad A_0 = 0.0 \qquad t_1 = 0.1 \qquad A_1 = 1.0$

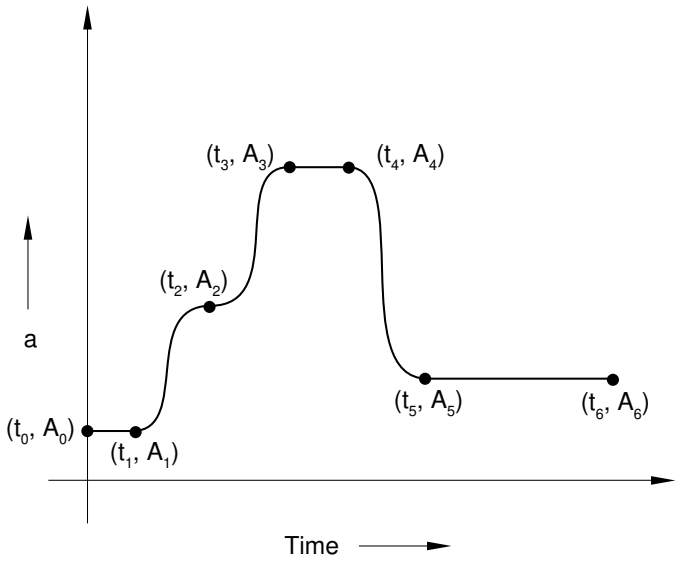
$a = A_0 \text{ for } t \le t_0$

$= A_0 + (A_1 - A_0)\,\xi^3\,(10 - 15\,\xi + 6\,\xi^2) \text{ for } t_0 < t < t_1$

$= A_1 \text{ for } t \ge t_1$

$\text{where } \xi = \dfrac{t - t_0}{t_1 - t_0}$

Figure 5: Smooth step amplitude definition example with two data points.



$t_0 = 0.0$ $A_0 = 0.1$ $t_1 = 0.1$ $A_1 = 0.1$ $t_2 = 0.2$ $A_2 = 0.3$ $t_3 = 0.3$ $A_3 = 0.5$

$t_4 = 0.4$ $A_4 = 0.5$ $t_5 = 0.5$ $A_5 = 0.2$ $t_6 = 0.8$ $A_6 = 0.2$

$$a = A_0 \text{ for } t \le t_0$$

$$= A_6 \text{ for } t \ge t_6$$

Amplitude, a, between any two consecutive data points (t_i, A_i) and (t_{i+1}, A_{i+1}) is

$$a = A_i + (A_{i+1} - A_i)\,\xi^3\,(10 - 15\xi + 6\,\xi^2)$$

$$\text{where } \xi = \frac{t - t_i}{t_{i+1} - t_i}$$

Figure 6: Smooth step amplitude definition example with multiple data points.

Input File Usage: **AMPLITUDE*, NAME=*name*, DEFINITION=SMOOTH STEP

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **Smooth step**

Defining a Solution-Dependent Amplitude for Superplastic Forming Analysis

Abaqus/Standard can calculate amplitude values based on a solution-dependent variable. Choose the solution-dependent definition method to create a solution-dependent amplitude curve. The data consist of an initial value, a minimum value, and a maximum value. The amplitude starts with the initial value and is then modified based on the progress of the solution, subject to the minimum and maximum values. The maximum value is typically the controlling mechanism used to end the analysis. This method is used with creep strain rate control for superplastic forming analysis (see *Rate-Dependent Plasticity: Creep and Swelling*).

Input File Usage: **AMPLITUDE*, NAME=*name*, DEFINITION=SOLUTION DEPENDENT

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **Solution dependent**

Defining the Bubble Load Amplitude for an Underwater Explosion

Two interfaces are available in Abaqus for applying incident wave loads (see *Incident Wave Loading due to External Sources*). For either interface bubble dynamics can be described using a model internal to Abaqus. A description of this built-in mechanical model and the parameters that define the bubble behavior are discussed in *Defining Bubble Loading for Spherical Incident Wave Loading*. The related theoretical details are described in *Loading due to an incident dilatational wave field*.

The preferred interface for incident wave loading due to an underwater explosion specifies bubble dynamics using the UNDEX charge property definition (see *Defining Bubble Loading for Spherical Incident Wave Loading*). The alternative interface for incident wave loading uses the bubble definition described in this section to define bubble load amplitude curves.

An example of the bubble amplitude definition with the following input data is shown in *Figure 7*.

$$K = 5.21 \times 10^7, \quad k = 9.0 \times 10^{-5}, \quad A = 0.18, \quad B = 0.185,$$
$$K_c = 8.396 \times 10^8, \quad \gamma = 1.27, \quad \rho_c = 1.5 \times 10^3, \quad m_c = 226.8,$$
$$d_I = 137.16, \quad \rho_f = 1.0 \times 10^3, \quad c_f = 1.5 \times 10^3, \quad \mathbf{n}_X = 0.0,$$
$$\mathbf{n}_Y = 0.0, \quad \mathbf{n}_Z = 1.0, \quad g = 9.8, \quad p_{atm} = 9.8 \times 10^4,$$
$$T_{final} = 0.555.$$

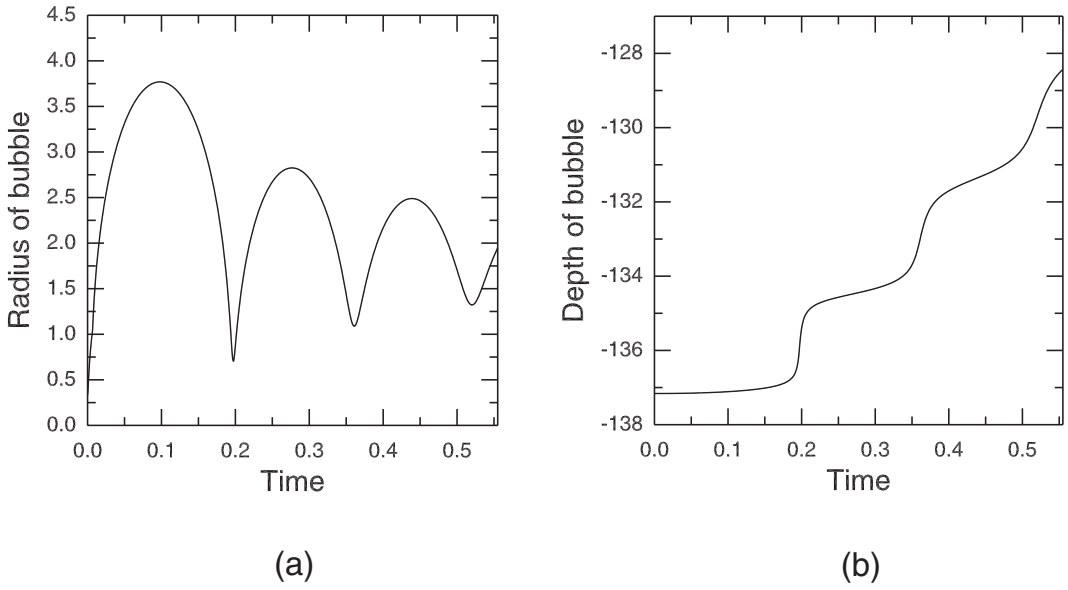


Figure 7: Bubble amplitude definition example: (a) radius of bubble and (b) depth of bubble center under fluid surface.

Input File Usage: **AMPLITUDE*, NAME=*name*, DEFINITION=BUBBLE

Abaqus/CAE Usage: Bubble amplitudes are not supported in Abaqus/CAE. However, bubble loading for an underwater explosion is supported in the Interaction module using the UNDEX charge property definition.

Defining an Amplitude via a User Subroutine

Choose the user definition method to define the amplitude curve via coding in user subroutine *UAMP* (Abaqus/Standard) or *VUAMP* (Abaqus/Explicit). You define the value of the amplitude function in time and, optionally, the values of the derivatives and integrals for the function sought to be implemented as outlined in *UAMP* and *VUAMP*.

You can use an arbitrary number of properties to calculate the amplitude, and you can use an arbitrary number of state variables that can be updated independently for each amplitude definition.

In Abaqus/Standard user-defined amplitudes are not supported for complex eigenvalue extraction, linear dynamic procedures, and steady-state dynamic analysis with the response computed directly in terms of the physical degrees of freedom.

Moreover, solution-dependent sensors can be used to define the user-customized amplitude. The sensors can be identified via their name, and two utilities allow for the extraction of the current sensor value inside the user subroutine (see *Obtaining Sensor Information*). Simple control/logical models can be implemented using this feature as illustrated in *Crank mechanism*.

Input File Usage: **AMPLITUDE*, NAME=*name*, DEFINITION=USER, PROPERTIES=*m*, VARIABLES=*n*

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **User**: **Number of variables**: *n*

User-defined amplitude properties are not supported in Abaqus/CAE.

Defining an Actuator Amplitude via Co-Simulation

The current value of an actuator amplitude can be imported at any given time from a co-simulation with a logical modeling program (see *About Co-Simulation*). The name specified on the actuator amplitude definition is used as the actuator name for co-simulation purposes. Therefore, at a given time each actuator is associated with one real number—the current value of the amplitude. As with any amplitude definition, the user-specified name can be used in conjunction with any Abaqus feature that can reference an amplitude.

Input File Usage: **AMPLITUDE*, NAME=*name*, DEFINITION=ACTUATOR

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **Actuator**

Using an Amplitude Definition with Boundary Conditions

When an amplitude curve is used to prescribe a variable of the model as a boundary condition (by referring to the amplitude from the boundary condition definition), the first and second time derivatives of the variable may also be needed. For example, the time history of a displacement can be defined for a direct integration dynamic analysis step by an amplitude variation; in this case Abaqus must compute the corresponding velocity and acceleration.

When the displacement time history is defined by a piecewise linear amplitude variation (tabular or equally spaced amplitude definition), the corresponding velocity is piecewise constant and the acceleration may be infinite at the end of each time interval given in the amplitude definition table, as shown in *Figure 8*(a). This behavior is unreasonable. (In Abaqus/Explicit time derivatives of amplitude curves are typically based on finite differences, such as $\frac{[A(t_{i+1})-A(t_i)]}{\Delta t}$, so there is some inherent smoothing associated with the time discretization.) You can modify the piecewise linear displacement variation into a combination of piecewise linear and piecewise quadratic variations through smoothing. Smoothing ensures that the velocity varies continuously during the time period of the amplitude definition and that the acceleration no longer has singularity points, as illustrated in *Figure 8*(b).

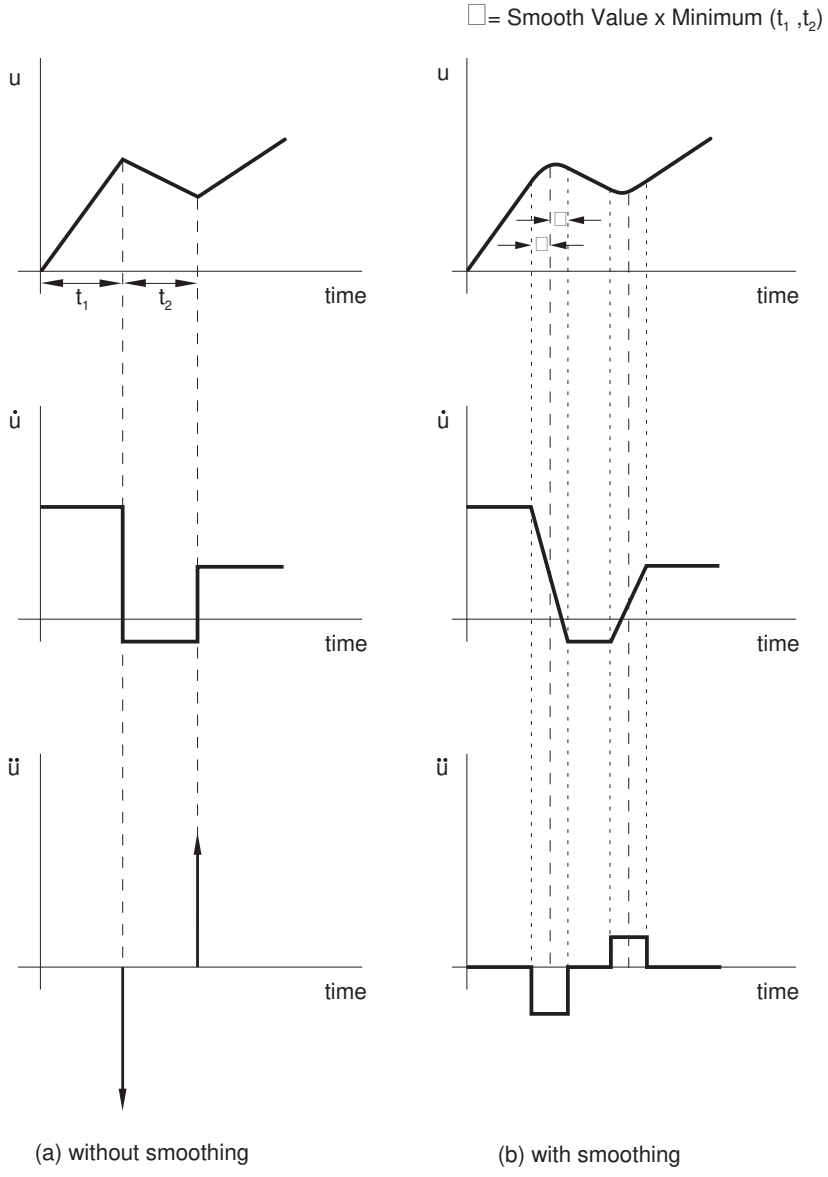


Figure 8: Piecewise linear displacement definitions.

When the velocity time history is defined by a piecewise linear amplitude variation, the corresponding acceleration is piecewise constant. Smoothing can be used to modify the piecewise linear velocity variation into a combination

of piecewise linear and piecewise quadratic variations. Smoothing ensures that the acceleration varies continuously during the time period of the amplitude definition.

You specify t, the fraction of the time interval before and after each time point during which the piecewise linear time variation is to be replaced by a smooth quadratic time variation. The default in Abaqus/Standard is t=0.25; the default in Abaqus/Explicit is t=0.0. The allowable range is $0.0 < t \leq 0.5$. A value of 0.05 is suggested for amplitude definitions that contain large time intervals to avoid severe deviation from the specified definition.

In Abaqus/Explicit if a displacement jump is specified using an amplitude curve (that is, the beginning displacement defined using the amplitude function does not correspond to the displacement at that time), this displacement jump will be ignored. Displacement boundary conditions are enforced in Abaqus/Explicit in an incremental manner using the slope of the amplitude curve. To avoid the "noisy" solution that may result in Abaqus/Explicit when smoothing is not used, it is better to specify the velocity history of a node rather than the displacement history (see *Boundary Conditions*).

When an amplitude definition is used with prescribed conditions that do not require the evaluation of time derivatives (for example, concentrated loads, distributed loads, temperature fields, etc., or a static analysis), the use of smoothing is ignored.

When the displacement time history is defined using a smooth-step amplitude curve, the velocity and acceleration will be zero at every data point specified, although the average velocity and acceleration may well be nonzero. Hence, this amplitude definition should be used only to define a (smooth) step function.

Input File Usage: Use either of the following options:

**AMPLITUDE*, NAME=*name*, DEFINITION=TABULAR, SMOOTH=*t*
**AMPLITUDE*, NAME=*name*, DEFINITION=EQUALLY SPACED, SMOOTH=*t*

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: choose **Tabular** or **Equally spaced**: **Smoothing: Specify:** *t*

Using an Amplitude Definition with Secondary Base Motion in Modal Dynamics

When an amplitude curve is used to prescribe a variable of the model as a secondary base motion in a modal dynamics procedure (by referring to the amplitude from the base motion definition during a modal dynamic procedure), the first or second time derivatives of the variable may also be needed. For example, the time history of a displacement can be defined for secondary base motion in a modal dynamics procedure. In this case Abaqus must compute the corresponding acceleration.

The modal dynamics procedure uses an exact solution for the response to a piecewise linear force. Accordingly, secondary base motion definitions are applied as piecewise linear acceleration histories. When displacement-type or velocity-type base motions are used to define displacement or velocity time histories and an amplitude variation using the tabular, equally spaced, periodic, modulated, or exponential decay definitions is used, an algorithmic acceleration is computed based on the tabular data (the amplitude data evaluated at the time values used in the modal dynamics procedure). At the end of any time increment where the amplitude curve is linear over that increment, linear over the previous increment, and the slopes of the amplitude variations over the two increments are equal, this algorithmic acceleration reproduces the exact displacement and velocity for displacement time histories or the exact velocity for velocity time histories.

When the displacement time history is defined using a smooth-step amplitude curve, the velocity and acceleration are zero at every data point specified, although the average velocity and acceleration may well be nonzero. Hence, this amplitude definition should be used only to define a (smooth) step function.

Defining Multiple Amplitude Curves

You can define any number of amplitude curves and refer to them from any load, boundary condition, or predefined field definition. For example, one amplitude curve can be used to specify the velocity of a set of nodes, while another amplitude curve can be used to specify the magnitude of a pressure load on the body. If the velocity and the pressure both follow the same time history, however, they can both refer to the same amplitude curve. There is one exception in Abaqus/Standard: only one solution-dependent amplitude (used for superplastic forming) can be active during each step.

Defining a Normalized Amplitude Curve

You can define a normalized amplitude and refer to it in multiple steps. The time values for the amplitude must be in the range of 0–1. The normalized time is multiplied internally by the current step time during the analysis. You can use a normalized amplitude with tabular and smooth step amplitude definitions.

For example, you can use a normalized amplitude with loads or boundary conditions that are applied similarly from step to step (such as ramping up to a constant value) but the step times are different. In this case you only need to define the normalized amplitude once and specify it in each of the steps.

Input File Usage: **AMPLITUDE*, NORMALIZED=YES

Abaqus/CAE Usage: Defining a normalized amplitude curve is not supported in Abaqus/CAE.

Scaling and Shifting Amplitude Curves

You can scale and shift both time and magnitude when defining an amplitude. This can be helpful, for example, when your amplitude data need to be converted to a different unit system or when you reuse existing amplitude data to define similar amplitude curves. If both scaling and shifting are applied at the same time, the amplitude values are first scaled and then shifted. The amplitude shifting and scaling can be applied to all amplitude definition types except for solution dependent, bubble, and user; for the actuator amplitude definition type, only scaling and shifting of the amplitude magnitude is supported.

Input File Usage: **AMPLITUDE*, NAME=*name*, SHIFTX=*shiftx_value*, SHIFTY=*shifty_value*, SCALEX=*scalex_value*, SCALEY=*scaley_value*

Abaqus/CAE Usage: The scaling and shifting of amplitude curves is not supported in Abaqus/CAE.

Reading the Data from an Alternate File

The data for an amplitude curve can be contained in a separate file.

Input File Usage: **AMPLITUDE*, NAME=*name*, INPUT=*file_name*

If the INPUT parameter is omitted, it is assumed that the data lines follow the keyword line.

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: any type: click mouse button 3 while holding the cursor over the data table, and select **Read from File**

Baseline Correction in Abaqus/Standard

When an amplitude definition is used to define an acceleration history in the time domain (a seismic record of an earthquake, for example), the integration of the acceleration record through time may result in a relatively large displacement at the end of the event. This behavior typically occurs because of instrumentation errors or a sampling frequency that is not sufficient to capture the actual acceleration history. In Abaqus/Standard it is possible to compensate for it by using "baseline correction."

The baseline correction method allows an acceleration history to be modified to minimize the overall drift of the displacement obtained from the time integration of the given acceleration. It is relevant only with tabular or equally spaced amplitude definitions.

Baseline correction can be defined only when the amplitude is referenced as an acceleration boundary condition during a direct-integration dynamic analysis or as an acceleration base motion in modal dynamics.

Input File Usage: Use both of the following options to include baseline correction:

**AMPLITUDE*, DEFINITION=TABULAR or EQUALLY SPACED
**BASELINE CORRECTION*

The **BASELINE CORRECTION* option must appear immediately following the data lines of the **AMPLITUDE* option.

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: choose **Tabular** or **Equally spaced**: **Baseline Correction**

Effects of Baseline Correction

The acceleration is modified by adding a quadratic variation of acceleration in time to the acceleration definition. The quadratic variation is chosen to minimize the mean squared velocity during each correction interval. Separate quadratic variations can be added for different correction intervals within the amplitude definition by defining the correction intervals. Alternatively, the entire amplitude history can be used as a single correction interval.

The use of more correction intervals provides tighter control over any "drift" in the displacement at the expense of more modification of the given acceleration trace. In either case, the modification begins with the start of the amplitude variation and with the assumption that the initial velocity at that time is zero.

The baseline correction technique is described in detail in *Baseline correction of accelerograms*.

Initial Conditions

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Prescribed Conditions*
- **INITIAL CONDITIONS*
- *Using the predefined field editors*

Overview

You can use initial conditions to prescribe the initial value of relevant quantities, such as solution variables, predefined fields, and material state. You specify initial conditions for particular nodes or elements, as appropriate. You can provide the data directly; in an external input file; or, in some cases, by a user subroutine or by the results or output database file from a previous Abaqus analysis.

If initial conditions are not specified, all initial conditions are zero except

- relative density in the porous metal plasticity model (which has the value 1.0);
- solution-dependent variables that control element deletion (which has the value 1.0); and
- element solution-dependent variables that control element deletion (which has the value 1.0).

Specifying Initial Conditions

You can specify initial conditions as follows:

- Directly in an input file.
- In an alternate input file (see *Reading the Input Data from an External File*).
- In a user subroutine.
- From the results or output database file of a previous Abaqus analysis. For more details on reading from output databases in the `.odb` or `.sim` file format, see *Importing Data from an Output Database File* and *Initial Condition Types*.

You can specify various types of initial conditions, depending on the analysis to be performed. For information about each type of initial condition, see *Initial Condition Types*.

Reading the Input Data from an External File

The input data for an initial conditions definition can be contained in a separate file. See *Input Syntax Rules* for the syntax of such file names.

Input File Usage: **INITIAL CONDITIONS*, INPUT=*file_name*

Abaqus/CAE Usage: Initial conditions cannot be read from a separate file in Abaqus/CAE.

Importing Data from an Output Database File

You can use field results from a prior simulation as initial conditions for another simulation in many cases. This method uses functionality associated with importing external fields, discussed in *General Capability for Importing External Fields*. The output database from the prior simulation must be in the `.sim` format. If the previous analysis is performed with third-party software, you must convert the results file to the `.sim` file format.

This method for specifying initial conditions cannot be used for initial acoustic static pressure, initial location of an enriched feature, initial fluid pressure, initial gap, initial mass flow rate, reference mesh (initial metric) for membrane elements, initial velocities specified either directly or in terms of an angular velocity and a global translational velocity, initial spud embedment, initial spud preload, initial unfold coordinates, and initial volume fraction.

You can use any appropriate result from the output database by specifying an output variable identifier (see *Abaqus/Standard Output Variable Identifiers* and *Abaqus/Explicit Output Variable Identifiers*, for available output variable identifiers). The results data can be associated with nodes or elements.

You can specify a source region (node or element set in the previous model) if data are imported only from a subset of the previous model. Sometimes a source region is also specified to eliminate ambiguity during mapping. You can specify a target region if data are specified only on a subset of the current model.

You must specify the full name of the output database file including the file extension `.sim`.

You can import only results data requested on two- and three-dimensional continuum elements and three-dimensional conventional shell elements, and you can specify imported data only for elements with matching types. You cannot import data from three-dimensional continuum elements to shell elements, or vice versa. When importing tensor field data, the source region must contain only elements with the same number of components of the tensor field. For example, when specifying initial stress with stress data from a previous analysis, you must separate the solid elements and shell elements in the previous analysis into separate source regions.

When you use results data to specify some initial conditions, data are mapped directly to all integration points and section points of a target element. These conditions include initial stress, initial plastic strain, initial damage initiation, initial hardening, initial specific energy, and initial maximum energy density used in Mullins effect models.

The following considerations pertain to importing data from shell elements to shell elements:

- You can import data from shell elements to shell elements with a different number of section points, except for initial temperatures and initial field variables.
- When importing element data, Abaqus requires that the elements in the same source/target region have the same number of layers, section points, and integration rules.
- When importing element data between shell elements with multiple layers, Abaqus requires the source and target elements to have the same number of layers. If results data are found only at the default output points, the same shell thickness between source and target elements is also required. Abaqus also requires that the elements in the same source/target region have the same shell thickness for each layer in this case.
- When results data are requested at all section points in the previous analysis, if a different number of section points are detected in the source and target elements, Abaqus automatically interpolates linearly in the thickness direction between the two closest section points in the source element to find the value at the section point in the target element. When results data are requested only at the default output points in the previous analysis, Abaqus interpolates linearly in the thickness direction using data at the top and bottom of the source shell elements.

You can specify mapping tolerances and special tensor averaging methods if mapping is performed. If the model in the previous analysis is repositioned in the current analysis, you must specify the translation and rotation so that the source region can be repositioned before data are imported, except in the following cases;

- Scalar data are imported from a matching mesh.
- Tensor data are imported from a matching mesh, and there is no rotation between the source region and the target region.

Input File Usage: Use the following options to define initial conditions by importing data from a source region in an output database (`.sim`) file to a target region in the current analysis. Mapping controls and repositioning are also applied:

**INITIAL CONDITIONS*
**EXTERNAL FIELD*, FILE=*file name*
blank, *target_set*, *blank*, NODES/ELEMENTS, *source_set*, *outputvar*, *mapcontrol*, *fieldop*

**FIELD MAPPER CONTROLS*, NAME=*mapcontrol*
, , 1.0

**FIELD OPERATIONS*, NAME=*fieldop*
x, *y*, *z*
x1, *y1*, *z1*, *x2*, *y2*, *z2*, *rotation*

Abaqus/CAE Usage: Defining initial conditions by importing field data from an output database (`.sim`) file is not supported in Abaqus/CAE.

Consistency with Kinematic Constraints

Abaqus does not ensure that initial conditions are consistent with multi-point or equation constraints for nodal quantities other than velocity (see *General Multi-Point Constraints* and *Linear Constraint Equations*). Initial conditions on nodal quantities such as temperature in heat transfer analysis, pore pressure in soils analysis, or acoustic pressure in acoustic analysis must be prescribed to be consistent with any multi-point constraint or equation constraint governing these quantities.

Spatial Interpolation Method

When you define initial conditions using a method that interpolates between dissimilar meshes, Abaqus operates by interpolating results from nodes in the old mesh to nodes in the new mesh. For each node:

1. The element (in the old mesh) in which the node lies is found, and the node's location in that element is obtained. (This procedure assumes that all nodes in the new mesh lie within the bounds of the old mesh: warning messages are issued if this is not so.)
2. The initial condition values are then interpolated from the nodes of the element (in the old mesh) to the new node.

Elements that do not support spatial interpolation include the complete libraries of convective heat transfer elements, axisymmetric elements with nonlinear axisymmetric deformation, axisymmetric surface elements,

truss elements, beam elements, link elements, hydrostatic fluid elements, solid infinite stress elements, and coupled thermal/electrical elements. Other specific elements that are not supported include: GKPS6, GKPE6, GKAX6, GK3D18, GK3D12M, GK3D4L,GK3D6L, GKPS4N, GKAX6N, GK3D18N, GK3D12MN, GK3D4LN, and GK3D6LN.

Initial Condition Types

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Prescribed Conditions*
- *Initial Conditions*
- **INITIAL CONDITIONS*
- *Using the predefined field editors*

Overview

This page describes the types of initial conditions that you can specify.

Defining Initial Acoustic Static Pressure

In Abaqus/Explicit you can define initial acoustic static pressure values at the acoustic nodes. These values should correspond to static equilibrium and cannot be changed during the analysis. You can specify the initial acoustic static pressure at two reference locations in the model, and Abaqus/Explicit interpolates these data linearly to the acoustic nodes in the specified node set. The linear interpolation is based on the projected position of each node onto the line defined by the two reference nodes. If the value at only one reference location is given, the initial acoustic static pressure is assumed to be uniform. The initial acoustic static pressure is used only in the evaluation of the cavitation condition (see *Acoustic Medium*) when the acoustic medium is capable of undergoing cavitation.

Input File Usage: **INITIAL CONDITIONS*, TYPE=ACOUSTIC STATIC PRESSURE

Abaqus/CAE Usage: Initial acoustic static pressure is not supported in Abaqus/CAE.

Defining Initial Volume Fraction of Material

You can prescribe the initial volume fraction of material in an element in analyses with progressive element activation (see *Progressive Element Activation*). At the beginning of an analysis the element must be either inactive or fully active; therefore, the value of the initial volume fraction must be equal to zero or one.

Input File Usage: **INITIAL CONDITIONS*, TYPE=ACTIVATION

Abaqus/CAE Usage: Initial volume fraction of material is not supported in Abaqus/CAE.

Defining Initial Volume Fraction of Material by Importing Field Data from an Output Database File

For three-dimensional continuum elements, you can define the initial volume fraction of material by importing field data as initial values of volume fraction at a particular step and increment or a user-specified time from the output database (`.sim`) file of a previous analysis. For more information, see *Importing Data from an Output Database File*.

Input File Usage: Use the following options to define initial values of the volume fraction of material using output from an output database (`.sim`) file:

**INITIAL CONDITIONS*, TYPE=ACTIVATION
**EXTERNAL FIELD*, FILE=*file name*

Abaqus/CAE Usage: Defining the initial volume fraction of material by importing field data from an output database (`.sim`) file is not supported in Abaqus/CAE.

Defining Initial Normalized Concentration

In Abaqus/Standard you can define initial normalized concentration values for use with diffusion elements in mass diffusion analysis (see *Mass Diffusion Analysis*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=CONCENTRATION

Abaqus/CAE Usage: Initial normalized concentration is not supported in Abaqus/CAE.

Defining Initially Bonded Contact Surfaces

In Abaqus/Standard you can define initially bonded or partially bonded contact surfaces. This type of initial condition is intended for use with the crack propagation capability (see *Crack Propagation Analysis*). The surfaces specified have to be different; this type of initial condition cannot be used with self-contact.

If the crack propagation capability is not activated, the bonded portion of the surfaces do not separate. In this case defining initially bonded contact surfaces would have the same effect as defining tied contact, which generates a permanent bond between two surfaces during the entire analysis (*Defining Tied Contact in Abaqus/Standard*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=CONTACT

Abaqus/CAE Usage: Initially bonded surfaces are not supported in Abaqus/CAE.

Defining Initial Damage Initiation

You can define initial values for the damage initiation measure for the ductile, shear, and the Müschenborn and Sonne forming limit diagram based damage initiation criteria (*Damage Initiation for Ductile Metals*). This capability is particularly useful in situations where a metal forming operation is carried out in one analysis, which is followed by a separate analysis that subjects the formed metal part to further deformation. The damage initiation measures at the end of the first analysis can be directly specified as initial conditions for the second analysis.

An alternate but approximate way of modeling initial conditions on damage initiation is by specifying the initial values of the equivalent plastic strain. Abaqus computes damage initiation measures based on the specified initial equivalent plastic strain, assuming a linear strain path between the initial (undeformed) state and the final (deformed) state. This approximation does not work well for deformation paths that deviate significantly from linearity in the strain space.

Input File Usage: Use the following option to specify the damage initiation measure for the ductile damage initiation criterion:

**INITIAL CONDITIONS*, TYPE=DAMAGE INITIATION, CRITERION=DUCTILE

Use the following option to specify the damage initiation measure for the shear damage initiation criterion:

**INITIAL CONDITIONS*, TYPE=DAMAGE INITIATION, CRITERION=SHEAR

Use the following option to specify the damage initiation measure for the Müschenborn and Sonne forming limit diagram–based damage initiation criterion:

**INITIAL CONDITIONS*, TYPE=DAMAGE INITIATION, CRITERION=MSFLD

Abaqus/CAE Usage: Defining initial values for the damage initiation measures is not supported in Abaqus/CAE.

Defining Initial Damage Initiation for Rebars

Initial values for damage initiation can also be defined for rebars within elements for the ductile and shear damage initiation criteria (see *Defining Rebar as an Element Property*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=DAMAGE INITIATION, REBAR

Abaqus/CAE Usage: Initial damage initiation for rebars is not supported in Abaqus/CAE.

Defining Initial Damage Initiation That Varies through the Thickness of Shell Elements

Initial values of damage initiation can be defined at each section point through the thickness of shell elements for the ductile and shear damage initiation criteria.

Input File Usage: **INITIAL CONDITIONS*, TYPE=DAMAGE INITIATION, SECTION POINTS

Abaqus/CAE Usage: Defining initial damage initiation that varies through the thickness of shell elements is not supported in Abaqus/CAE.

Defining Initial Damage Initiation by Importing Field Data from an Output Database File

For three-dimensional continuum elements, you can define initial ductile and shear damage initiation criteria by importing field data as initial values of damage initiation at a particular step and increment or a user-specified time in the output database (`.sim`) file of a previous analysis. For more information, see *Importing Data from an Output Database File*.

Input File Usage: Use the following options to define initial values of ductile damage initiation using ductile damage initiation criterion output from an output database (`.sim`) file:

**INITIAL CONDITIONS*, TYPE=DAMAGE INITIATION, CRITERION=DUCTILE
**EXTERNAL FIELD*, FILE=*file name*

Use the following options to define initial values of shear damage initiation using shear damage initiation criterion output from a user-specified output database (`.sim`) file.

**INITIAL CONDITIONS*, TYPE=DAMAGE INITIATION, CRITERION=SHEAR
**EXTERNAL FIELD*, FILE=*file name*

Abaqus/CAE Usage: Defining initial damage initiation by importing field data from an output database (`.sim`) file is not supported in Abaqus/CAE.

Defining the Initial Location of an Enriched Feature

You can specify the initial location of an enriched feature, such as a crack, in an Abaqus/Standard analysis (see *Modeling Discontinuities as an Enriched Feature Using the Extended Finite Element Method*). Two signed distance functions per node are generally required to describe the crack location, including the location of crack tips, in a cracked geometry. The first signed distance function describes the crack surface, while the second is used to construct an orthogonal surface such that the intersection of the two surfaces defines the crack front. The first signed distance function is assigned only to nodes of elements intersected by the crack, while the second is assigned only to nodes of elements containing the crack tips. No explicit representation of the crack is needed because the crack is entirely described by the nodal data.

Input File Usage: **INITIAL CONDITIONS*, TYPE=ENRICHMENT

Abaqus/CAE Usage: Interaction module: crack editor: **Crack location**: **Specify**: select region

Defining Initial Values of Element Solution-Dependent Variables

You can define initial values of solution-dependent state variables (see *About User Subroutines and Utilities*). The initial values can be defined directly.

Input File Usage: **INITIAL CONDITIONS*, TYPE=ESDV

Abaqus/CAE Usage: Initial element solution-dependent variables are not supported in Abaqus/CAE.

Defining Initial Values of Element Solution-Dependent Variables from an Output Database File

For three-dimensional continuum elements, you can define initial values of element solution-dependent variables by importing field data as initial values of element solution-dependent variables at a particular step and increment or a user-specified time in the output database (`.sim`) file of a previous analysis. For more information, see *Importing Data from an Output Database File*.

Input File Usage:
**INITIAL CONDITIONS*, TYPE=ESDV
**EXTERNAL FIELD*, FILE=*file name*
, , ESDV*i*, , , ESDV*i*
…
, , ESDV*n*, , , ESDV*n*

Abaqus/CAE Usage: Defining initial element solution-dependent variables by importing field data from an output database (`.sim`) file is not supported in Abaqus/CAE.

Defining Initial Values of Predefined Field Variables

You can define initial values of predefined field variables. The values can be changed during an analysis (see *Predefined Fields*).

You must specify the field variable number being defined, *n*. Any number of field variables can be used; each must be numbered consecutively (1, 2, 3, etc.). Repeat the initial conditions definition, with a different field variable number, to define initial conditions for multiple field variables. The default is *n*=1.

The definition of initial field variable values must be compatible with the section definition and with adjacent elements, as explained in *Predefined Fields*.

Input File Usage: **INITIAL CONDITIONS*, TYPE=FIELD, VARIABLE=*n*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**, choose **Other** for the **Category** and **Field** for the **Types for Selected Step**; select region; **Field variable number**: *n*

Defining Uniform Initial Fields

You can apply uniform initial field variables to either the entire model or to node sets that you specify. Omit the node number or node set to apply the specified field variable to all nodes in the model automatically.

You can specify uniform field variables with all element types, including beams and shells. The specified uniform field is applied to all section points in beams and shells. However, the definition of initial field variables must be compatible with the section definition of the element and with adjacent elements, as explained in *Predefined Fields*. Abaqus issues a warning message during input file preprocessing if an initial field variable is applied to any node that is associated with at least one shell or beam section that specifies field variables using gradients and at least one section that directly specifies the values of the field variables.

Input File Usage: **INITIAL CONDITIONS*, TYPE=FIELD, VARIABLE=*n*, SECTION SPECIFICATION=UNIFORM

Abaqus/CAE Usage: Defining uniform initial fields is not supported in Abaqus/CAE.

Initializing Predefined Field Variables with Nodal Temperature Records from a User-Specified Results File

You can define initial values of predefined field variables using nodal temperature records from a particular step and increment of a results file from a previous Abaqus analysis or from a results file you create (see *Predefined Fields*). The previous analysis is most commonly an Abaqus/Standard heat transfer analysis. The use of the `.fil` file extension is optional.

The part (`.prt`) file from the previous analysis is required to read the initial values of predefined field variables from the results file (*Assembly Definition*). Both the previous model and the current model must be consistently defined in terms of an assembly of part instances.

Input File Usage: **INITIAL CONDITIONS*, TYPE=FIELD, VARIABLE=*n*, FILE=*file*, OUTPUT VARIABLE=NT, STEP=*step*, INC=*inc*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**, choose **Other** for the **Category** and **Field** for the **Types for Selected Step**; select region; **Field variable number**: *n*, **File name**: *file*, **Step**: *step*, **Increment**: *inc*

Defining Initial Predefined Field Variables Using Scalar Nodal Output from a User-Specified Output Database File

You can define initial values of predefined field variables using scalar nodal output variables from a particular step and increment in the output database file of a previous Abaqus/Standard analysis. For a list of scalar nodal output variables that can be used to initialize a predefined field, see *Predefined Fields*.

The part (`.prt`) file from the previous analysis is required to read initial values from the output database file (see *Assembly Definition*). Both the previous model and the current model must be defined consistently in terms of an assembly of part instances; node numbering must be the same, and part instance naming must be the same.

The file extension is optional; however, only the output database file can be used for this option.

Input File Usage: **INITIAL CONDITIONS*, TYPE=FIELD, VARIABLE=*n*, FILE=*file*, OUTPUT VARIABLE=*scalar nodal output variable*, STEP=*step*, INC=*inc*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**, choose **Other** for the **Category** and **Field** for the **Types for Selected Step**; select region; **Field variable number**: *n*, **File name**: *file*, **Output variable**: *scalar nodal output variable*, **Step**: *step*, **Increment**: *inc*

Defining Initial Predefined Field Variables by Interpolating Scalar Nodal Output Variables for Dissimilar Meshes from a User-Specified Output Database File

When the mesh for one analysis is different from the mesh for the subsequent analysis, Abaqus can interpolate scalar nodal output variables (using the undeformed mesh of the original analysis) to predefined field variables that you choose. For a list of supported scalar nodal output variables that can be used to define predefined field variables, see *Predefined Fields*. This technique can also be used in cases where the meshes match but the node number or part instance naming differs between the analyses. Abaqus looks for the `.odb` extension automatically. The part (`.prt`) file from the previous analysis is required if that analysis model is defined in terms of an assembly of part instances (see *Assembly Definition*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=FIELD, VARIABLE=*n*,
OUTPUT VARIABLE=*scalar nodal output variable*,
INTERPOLATE, FILE=*file*, STEP=*step*, INC=*inc*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**, choose **Other** for the **Category** and **Field** for the **Types for Selected Step**; select region; **Field variable number**: *n*, **File name**: *file*, **Output variable**: *scalar nodal output variable*, **Step**: *step*, **Increment**: *inc*, **Mesh compatibility**: **Incompatible**

Defining Initial Predefined Field Variables by Importing Field Data from an Output Database File

For three-dimensional continuum elements, you can define initial predefined field variables by importing field data at a particular step and increment or a user-specified time in the output database (`.sim`) file of a previous analysis. For more information, see *Importing Data from an Output Database File*.

Input File Usage: Use the following options to define initial values of predefined field variables using output variables from an output database (`.sim`) file:

**INITIAL CONDITIONS*, TYPE=FIELD, VARIABLE=*n*,
**EXTERNAL FIELD*, FILE=*file name*

Abaqus/CAE Usage: Defining initial predefined field variables by importing field data from an output database (`.sim`) file is not supported in Abaqus/CAE.

Defining Initial Predefined Pore Fluid Pressure

You can associate a known (precomputed) pore fluid pressure field with a specific predefined field variable (as discussed in *Predefined Fields*) and read the field into a static or an explicit dynamic stress analysis. To initialize the pore fluid pressure field, you must initialize the corresponding predefined field variable.

Input File Usage: Use the following option to define initial values of predefined pore fluid pressure, where it is assumed that field variable *n* is associated with the known pore fluid pressure field:

**INITIAL CONDITIONS*, TYPE=FIELD, VARIABLE=*n*

Abaqus/CAE Usage: Defining initial predefined pore fluid pressure is not supported in Abaqus/CAE.

Defining Initial Fluid Electric Potential

In Abaqus/Standard you can define initial fluid electric potential values for use with coupled thermal-electrochemical elements in a coupled thermal-electrochemical analysis (see *Coupled Thermal-Electrochemical Analysis*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=FLUID ELECTRIC POTENTIAL

Abaqus/CAE Usage: Initial fluid electric potential is not supported in Abaqus/CAE.

Defining Initial Fluid Pressure in Fluid-Filled Structures

You can prescribe initial pressure for fluid-filled structures (see *About Surface-Based Fluid Cavities*).

Do not use this type of initial condition to define initial conditions in porous media in Abaqus/Standard; use initial pore fluid pressures instead (see below).

Input File Usage: **INITIAL CONDITIONS*, TYPE=FLUID PRESSURE

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**, choose **Other** for the **Category** and **Fluid cavity pressure** for the **Types for Selected Step**; select a fluid cavity interaction; **Fluid cavity pressure**: *pressure*

Defining Initial Values of State Variables for Plastic Hardening

You can prescribe initial equivalent plastic strain and, if relevant, the initial backstress tensor for elements that use one of the metal plasticity (*Inelastic Behavior*) or Drucker-Prager (*Extended Drucker-Prager Models*) material models. These initial quantities are intended for materials in a work hardened state; they can be defined directly or by user subroutine *HARDINI*. You can also prescribe initial values for the volumetric compacting plastic strain, $-\varepsilon_{\text{vol}}^{pl}$, for elements that use the crushable foam material model with volumetric hardening (*Crushable Foam Plasticity Models*).

You can also specify multiple backstresses for the nonlinear kinematic hardening model. Optionally, you can specify the kinematic shift tensor (backstress) using the full tensor format, regardless of the element type to which the initial conditions are applied.

Input File Usage: **INITIAL CONDITIONS*, TYPE=HARDENING, NUMBER BACKSTRESSES=*n*, FULL TENSOR

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**, choose **Mechanical** for the **Category** and **Hardening** for the **Types for Selected Step**; select region; **Number of backstresses**: *n*

Defining Initial Equivalent Plastic Strain by Importing Field Data from an Output Database File

For three-dimensional continuum elements, you can define initial equivalent plastic strain by importing field data as equivalent plastic strain at a particular step and increment or a user-specified time in the output database (`.sim`) file of a previous analysis. For more information, see *Importing Data from an Output Database File*.

Input File Usage: **INITIAL CONDITIONS*, TYPE=HARDENING
**EXTERNAL FIELD*, FILE=*file name*

Abaqus/CAE Usage: Defining initial equivalent plastic strain by importing field data from an output database (`.sim`) file is not supported in Abaqus/CAE.

Defining Initial Equivalent Plastic Strain and Backstress Tensor by Importing Field Data from an Output Database File

For three-dimensional continuum elements, you can define initial equivalent plastic strain and backstress tensor by importing field data as equivalent plastic strain and backstress tensor at a particular step and increment or a user-specified time in the output database (`.sim`) file of a previous analysis. For more information, see *Importing Data from an Output Database File*.

Input File Usage:
**INITIAL CONDITIONS*, TYPE=HARDENING, NUMBER BACKSTRESSES=*n*
**EXTERNAL FIELD*, FILE=*file name*
, , PEEQ, , , PEEQ
, , ALPHA*i*, , , ALPHA*i*
…
, , ALPHA*n*, , , ALPHA*n*

Abaqus/CAE Usage: Defining initial equivalent plastic strain and backstress tensor by importing field data from an output database (`.sim`) file is not supported in Abaqus/CAE.

Defining Hardening Parameters for Rebars

The hardening parameters can also be defined for rebars within elements. Rebars are discussed in *Defining Rebar as an Element Property*.

Input File Usage: **INITIAL CONDITIONS*, TYPE=HARDENING, REBAR

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**, choose **Mechanical** for the **Category** and **Hardening** for the **Types for Selected Step**; select region; **Definition: Rebar**

Defining Hardening Parameters in User Subroutine `HARDINI`

For complicated cases in Abaqus/Standard user subroutine *HARDINI* can be used to define the initial work hardening. In this case Abaqus/Standard calls the subroutine at the start of the analysis for each material point in the model. You can then define the initial conditions at each point as a function of coordinates, element number, etc.

Input File Usage: **INITIAL CONDITIONS*, TYPE=HARDENING, USER

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**, choose **Mechanical** for the **Category** and **Hardening** for the **Types for Selected Step**; select region; **Definition: User-defined**

Defining Elements Initially Open for Tangential Fluid Flow

You can specify the pore pressure cohesive elements that are initially open for tangential fluid flow (see *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*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=INITIAL GAP

Abaqus/CAE Usage: Initial gap is not supported in Abaqus/CAE.

Defining Initial Slurry Concentration

In Abaqus/Standard you can define initial slurry concentration values for use with an analysis that models slurry transport and placement.

Input File Usage: **INITIAL CONDITIONS*, TYPE=SLURRYVF

Abaqus/CAE Usage: Initial slurry concentration is not supported in Abaqus/CAE.

Defining Initial Ion Concentration

In Abaqus/Standard you can define initial ion concentration values for use with coupled thermal-electrochemical elements in a coupled thermal-electrochemical analysis (see *Coupled Thermal-Electrochemical Analysis*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=ION CONCENTRATION

Abaqus/CAE Usage: Initial ion concentration is not supported in Abaqus/CAE.

Defining Initial Species Concentration

In Abaqus/Standard you can define initial species concentration values at the cathode that is modeled using coupled thermal-electrochemical elements in an analysis that uses solid electrolytes (see *Modeling Solid Electrolytes and Solid-State Batteries*). "Species" refers to the chemical species that participates in the primary electrochemical reactions in the battery.

Input File Usage: **INITIAL CONDITIONS*, TYPE=SPECIES CONCENTRATION

Abaqus/CAE Usage: Initial species concentration is not supported in Abaqus/CAE.

Defining Initial Mass Flow Rates in Forced Convection Heat Transfer Elements

In Abaqus/Standard you can define the initial mass flow rate through forced convection heat transfer elements. You can specify a predefined mass flow rate field to vary the value of the mass flow rate within the analysis step (see *Uncoupled Heat Transfer Analysis*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=MASS FLOW RATE

Abaqus/CAE Usage: Initial mass flow rate is not supported in Abaqus/CAE.

Defining Initial Maximum Energy Density in Mullins Effect Models

You can define values of the initial maximum energy density used in Mullins effect models. This maximum energy density is the maximum deviatoric strain energy density for rubberlike materials and the maximum strain energy density for elastomeric foams (see *Stress Softening in Elastomers*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=MULLINS

Abaqus/CAE Usage: Initial maximum Mullins energy density is not supported in Abaqus/CAE.

Defining Initial Values of Plastic Strain

You can define an initial plastic strain field on elements that use one of the metal plasticity (*Inelastic Behavior*), critical state (clay) plasticity (*Critical State (Clay) Plasticity Model*), Drucker-Prager (*Extended Drucker-Prager Models*), or soft rock plasticity models. The specified plastic strain values are applied uniformly over the element unless they are defined at each section point through the thickness in shell elements.

If a local coordinate system is defined (see *Orientations*), the plastic strain components must be given in the local system.

Input File Usage: **INITIAL CONDITIONS*, TYPE=PLASTIC STRAIN

Abaqus/CAE Usage: Initial plastic strain conditions are not supported in Abaqus/CAE.

Defining Initial Plastic Strains for Rebars

Initial values of plastic strains can also be defined for rebars within elements (see *Defining Rebar as an Element Property*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=PLASTIC STRAIN, REBAR

Abaqus/CAE Usage: Initial plastic strain conditions are not supported in Abaqus/CAE.

Defining Initial Plastic Strains by Importing Field Data from an Output Database File

For three-dimensional continuum elements, you can define initial plastic strains by importing field data as initial plastic strains at a particular step and increment or a user-specified time in the output database (`.sim`) file of a previous analysis. For more information, see *Importing Data from an Output Database File*.

Input File Usage: **INITIAL CONDITIONS*, TYPE=PLASTIC STRAIN
**EXTERNAL FIELD*, FILE=*file name*

Abaqus/CAE Usage: Defining initial plastic strains by importing field data from an output database (`.sim`) file is not supported in Abaqus/CAE.

Defining Initial Pore Fluid Pressures in a Porous Medium

In Abaqus/Standard you can define the initial pore pressure, u_w, for nodes in a coupled pore fluid diffusion/stress analysis (see *Coupled Pore Fluid Diffusion and Stress Analysis*). The initial pore pressure can be defined either directly as an elevation-dependent function or by user subroutine *UPOREP*.

In Abaqus/Explicit you can define the initial pore pressure, u_w, for nodes in an undrained pore fluid flow and stress analysis (see *Undrained Pore Fluid Flow and Stress Analysis*). The initial pore pressure should be defined directly as an elevation-dependent function.

The initial effective stress/pore pressure conditions defined for an element are assumed to act on the initial configuration of the element. If the initial effective stress/pore pressure conditions are removed during the step, the element returns to a stress-free configuration that is different from the initial one. Since displacements and total strain output are measured relative to the initial configuration, the stress-free configuration will have nonzero values for the displacement and total strain fields that will depend on the initial conditions. While it is easy to verify the above behavior analytically in a one-element problem subjected to an initial stress and pore pressure field, the situation in a complex boundary value problem is determined by other factors that may make it difficult to resolve analytically.

Elevation-Dependent Initial Pore Pressures

When an elevation-dependent pore pressure is prescribed for a particular node set, the pore pressure in the vertical direction (assumed to be the z-direction in three-dimensional and axisymmetric models and the y-direction in two-dimensional models) is assumed to vary linearly with this vertical coordinate. You must give two pairs of pore pressure and elevation values to define the pore pressure distribution throughout the node set. Enter only the first pore pressure value (omit the second pore pressure value and the elevation values) to define a constant pore pressure distribution.

Input File Usage: **INITIAL CONDITIONS*, TYPE=PORE PRESSURE

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Other** for the **Category** and **Pore pressure** for the **Types for Selected Step**; select region; **Point 1 distribution: Uniform** or select an analytical field

Defining Initial Pore Pressures in User Subroutine UPOREP

For complicated cases initial pore pressure values can be defined by user subroutine *UPOREP*. In this case Abaqus/Standard makes a call to subroutine *UPOREP* at the start of the analysis for all nodes in the model. You can define the initial pore pressure at each node as a function of coordinates, node number, etc.

Input File Usage: **INITIAL CONDITIONS*, TYPE=PORE PRESSURE, USER

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Other** for the **Category** and **Pore pressure** for the **Types for Selected Step**; select region; **Point 1 distribution: User-defined**

Defining Initial Pore Pressure Values Using Nodal Pore Pressure Output from a User-Specified Output Database File

In Abaqus/Standard, you can define initial pore pressure values using nodal pore pressure output variables from a particular step and increment in the output database (`.odb`) file of a previous Abaqus/Standard analysis. The file extension is optional; however, only the output database file can be used.

The part (`.prt`) file from the previous analysis is required to read the initial values of predefined pore pressure from the results file (*Assembly Definition*). For the same mesh pore pressure mapping, both the previous model and the current model must be defined consistently, including node numbering, which must be the same in both models. If the models are defined in terms of an assembly of part instances, the part instance naming must be the same.

Input File Usage: **INITIAL CONDITIONS*, TYPE=PORE PRESSURE, FILE=*file*, STEP=*step*, INC=*inc*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Other** for the **Category** and **Pore pressure** for the **Types for Selected Step**; select region; **Point 1 distribution: From output database file**

Interpolating Initial Pore Pressure Values for Dissimilar Pore Pressure Mapping Values in a User-Specified Output Database File

For dissimilar mesh pore pressure mapping, interpolation is required. In Abaqus/Standard, you can also limit the interpolation region by specifying the source region in the form of an element set from which pore pressure is to be interpolated and the target region in the form of a node set onto which the pore pressure is mapped.

Input File Usage: **INITIAL CONDITIONS*, TYPE=PORE PRESSURE, FILE=*file*, INTERPOLATE, STEP=*step*, INC=*inc*
**INITIAL CONDITIONS*, TYPE=PORE PRESSURE, FILE=*file*, INTERPOLATE, STEP=*step*, INC=*inc*, DRIVING ELSETS

Abaqus/CAE Usage: You cannot specify the regions where pore pressure values are to be interpolated in Abaqus/CAE.

Defining Initial Pressure Stress in a Mass Diffusion Analysis

In Abaqus/Standard you can specify the initial pressure stress, $p \overset{\text{def}}{=} -\text{trace}\,(\boldsymbol{\sigma})/3$, at the nodes in a mass diffusion analysis (see *Mass Diffusion Analysis*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=PRESSURE STRESS

Abaqus/CAE Usage: Initial pressure stress is not supported in Abaqus/CAE.

Defining Initial Pressure Stress from a User-Specified Results File

You can define initial values of pressure stress as those values existing at a particular step and increment in the results file of a previous Abaqus/Standard stress/displacement analysis (see *Predefined Fields*). The use of the `.fil` file extension is optional. The part (`.prt`) file from the previous analysis is required to read the initial values of predefined pressure stress from the results file (*Assembly Definition*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=PRESSURE STRESS, FILE=*file*, STEP=*step*, INC=*inc*

Abaqus/CAE Usage: Initial pressure stress is not supported in Abaqus/CAE.

Defining Initial Void Ratios in a Porous Medium

In Abaqus/Standard you can specify the initial values of the void ratio, e, at the nodes of a porous medium (see *Coupled Pore Fluid Diffusion and Stress Analysis*). The initial void ratio can be defined either directly as an elevation-dependent function, by interpolation from a previous output database file, or by user subroutine *VOIDRI*.

In Abaqus/Explicit you can specify the initial values of the void ratio, e, at the nodes of a porous medium (see). The initial void ratio should be defined directly as an elevation-dependent function.

Elevation-Dependent Initial Void Ratio

When an elevation-dependent void ratio is prescribed for a particular node set, the void ratio in the vertical direction (assumed to be the z-direction in three-dimensional and axisymmetric models and the y-direction in two-dimensional models) is assumed to vary linearly with this vertical coordinate. When the void ratio is specified for a region meshed with fully integrated first-order elements, the nodal values of void ratio are interpolated to the centroid of the element and are assumed to be constant through the element. You must provide two pairs of void ratio and elevation values to define the void ratio throughout the node set. Enter only the first void ratio value (omit the second void ratio value and the elevation values) to define a constant void ratio distribution.

Input File Usage: **INITIAL CONDITIONS*, TYPE=RATIO

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Other** for the **Category** and **Void ratio** for the **Types for Selected Step**; select region; **Point 1 distribution: Uniform** or select an analytical field

Defining Void Ratio from a User-Specified Output Database

In Abaqus/Standard, you can define initial void ratios from the output database (`.odb`) file of a previous Abaqus/Standard soil analysis in which the void ratio is requested as output. The part (`.prt`) file from the previous analysis is required to read the initial values of predefined void ratios from the results file (*Assembly Definition*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=RATIO, FILE=*file*, STEP=*step*, INC=*inc*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Other** for the **Category** and **Void ratio** for the **Types for Selected Step**; select region; **Point 1 distribution: From output database file**

Interpolating Initial Void Ratios from Values in a User-Specified Output Database

In Abaqus/Standard, when you define initial void ratios from the output database (`.odb`) file of a previous Abaqus/Standard soil analysis, you can also limit the interpolation region by specifying the source region in the form of an element set from which void ratios are to be interpolated and the target region in the form of a node set onto which the void ratios are mapped. The part (`.prt`) file from the previous analysis is required to read the initial values of predefined void ratios from the results file (*Assembly Definition*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=RATIO,
INTERPOLATE, FILE=*file*, STEP=*step*, INC=*inc*, DRIVING ELSETS

Abaqus/CAE Usage: You cannot specify the regions where void ratios are to be interpolated in Abaqus/CAE.

Defining Void Ratios in User Subroutine `VOIDRI`

In Abaqus/Standard, for complicated cases initial values of the void ratios can be defined by user subroutine *VOIDRI*. In this case Abaqus/Standard makes a call to subroutine *VOIDRI* at the start of the analysis for each material integration point in the model. You can then define the initial void ratio at each point as a function of coordinates, element number, etc.

Input File Usage: **INITIAL CONDITIONS*, TYPE=RATIO, USER

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Other** for the **Category** and **Void ratio** for the **Types for Selected Step**; select region; **Point 1 distribution: User-defined**

Defining Initial Porosity in a Porous Medium

In Abaqus/Explicit you can specify the initial values of the porosity, n , at the element level of a porous medium (see).

Input File Usage: **INITIAL CONDITIONS*, TYPE=POROSITY

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Other** for the **Category** and **porosity** for the **Types for Selected Step**

Defining a Reference Mesh for Membrane Elements and Three-Dimensional Solid Elements

In Abaqus/Explicit you can specify a reference mesh (initial metric) for membrane elements and three-dimensional solid elements. For membrane elements, this is typically useful in airbag simulations to model the wrinkles that arise from the airbag folding process. A flat mesh might be suitable for the unstressed reference configuration, but the initial state might require a corresponding folded mesh defining the folded state. For solid elements, defining a reference mesh provides a convenient way to initialize stresses due to dummy positioning/settling in crash applications. Defining a reference configuration that is different from the initial configuration results in nonzero stresses in the initial configuration based on the elastic response of the material. For membrane elements, it can also result in nonzero strains relative to the reference configuration. For solids elements, initial strains are measured with respect to the initial configuration and are reported as zero. If a reference mesh is specified for an element, any initial stress conditions specified for the same element are ignored.

In the reference mesh method, the deformation gradient defined by the reference and initial configurations is used to update the stress state of the material in total form assuming purely elastic behavior. Therefore, the method is intended primarily for elastic or hyperelastic materials for which the stress state does not depend on deformation history. For plastic materials, the initial stresses computed by the reference mesh method might violate yield, and it might take several increments in an Abaqus/Explicit analysis for the stresses to settle and return back onto the yield surface. The reference mesh method provides a convenient way to initialize stresses in air bags, dummies, and other components in crash applications and provides an alternative solution to running an import analysis. For elements with internal nodes that you cannot access (such as C3D8I, C3D10M, and C3D10 elements), the initial stresses computed based on the user-defined reference mesh might not be the same as the initial stresses obtained from an import analysis. However, for most cases, the initial stresses obtained from these two methods should be reasonably close.

If rebar layers are defined in membrane elements, the angular orientation defined in the reference configuration is updated to obtain the same orientation in the initial configuration.

For membrane elements, you can define the reference mesh either with the element numbers and the coordinates of the nodes in each element or with the node numbers and node coordinates. For three-dimensional solid elements, you can only define the reference mesh using the node numbers and the corresponding node coordinates. For both methods, you must specify the coordinates of all the nodes in the element to have a valid initial condition for that element. The two methods are mutually exclusive.

If you define a reference mesh, the strain output is calculated based on the reference configuration for membrane elements; however, for three-dimensional solid elements, the strain output is based on the initial configuration. Therefore, you might observe nonzero initial strains and stresses for membrane elements and only nonzero initial stresses for three-dimensional solid elements.

Input File Usage: Use the following option to specify the reference mesh using element numbers and coordinates of all the element's nodes (applies only to membrane elements):

**INITIAL CONDITIONS*, TYPE=REF COORDINATE

Use the following option to specify the reference mesh using node numbers and the coordinates of the nodes (applies to membrane elements and three-dimensional solid elements):

**INITIAL CONDITIONS*, TYPE=NODE REF COORDINATE

Abaqus/CAE Usage: Specifying a reference mesh for membrane and three-dimensional solid elements is not supported in Abaqus/CAE.

Defining Initial Relative Density

You can specify the initial values of the relative density field for a porous metal plasticity material model (see *Porous Metal Plasticity*) or equations of state (see *Equation of State*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=RELATIVE DENSITY

Abaqus/CAE Usage: Initial relative density is not supported in Abaqus/CAE.

Defining Initial Angular and Translational Velocity

You can prescribe initial velocities in terms of an angular velocity and a translational velocity. This type of initial condition is typically used to define the initial velocity of a component of a rotating machine, such as a jet engine. The initial velocities are specified by giving the angular velocity, ω; the axis of rotation, defined from a point a at $\mathbf{X}^a$ to a point b at $\mathbf{X}^b$; and a translational velocity, $\mathbf{v}^g$. The initial velocity of node N at $\mathbf{X}^N$ is then

$$\mathbf{v}^N = \mathbf{v}^g + \omega \frac{\left(\mathbf{X}^b - \mathbf{X}^a\right)}{\left|\mathbf{X}^b - \mathbf{X}^a\right|} \times \left(\mathbf{X}^N - \mathbf{X}^a\right).$$

Input File Usage: **INITIAL CONDITIONS*, TYPE=ROTATING VELOCITY

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Mechanical** for the **Category** and **Velocity** for the **Types for Selected Step**

Defining Initial Saturation for a Porous Medium

In Abaqus/Standard you can define the initial saturation, s, for elements in a coupled pore fluid diffusion/stress analysis (see *Coupled Pore Fluid Diffusion and Stress Analysis*). If the porous material's absorption/exsorption behavior under partially saturated flow conditions is not defined, the initial saturation is set to 1.0 by default.

Input File Usage: **INITIAL CONDITIONS*, TYPE=SATURATION

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Other** for the **Category** and **Saturation** for the **Types for Selected Step**

Defining Initial Solid Electric Potential

In Abaqus/Standard you can define initial solid electric potential values for use with coupled thermal-electrochemical elements in a coupled thermal-electrochemical analysis (see *Coupled Thermal-Electrochemical Analysis*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=SOLID ELECTRIC POTENTIAL

Abaqus/CAE Usage: Initial solid electric potential is not supported in Abaqus/CAE.

Defining the Initial Values of Solution-Dependent State Variables

You can define initial values of solution-dependent state variables (see *About User Subroutines and Utilities*). The initial values can be defined directly or, in Abaqus/Standard, by user subroutine *SDVINI*. Values given directly are applied uniformly over the element.

Input File Usage: **INITIAL CONDITIONS*, TYPE=SOLUTION

Abaqus/CAE Usage: Initial solution-dependent variables are not supported in Abaqus/CAE.

Defining the Initial Values of Solution-Dependent State Variables by Importing Field Data from an Output Database File

For three-dimensional continuum elements, you can define initial values of solution-dependent state variables by importing field data as initial values of solution-dependent state variables at a particular step and increment or a user-specified time from the output database (`.sim`) file of a previous analysis. For more information, see *Importing Data from an Output Database File*.

Input File Usage: **INITIAL CONDITIONS*, TYPE=SOLUTION
**EXTERNAL FIELD*, FILE=*file name*
, , SDV*i*, , , SDV*i*
…
, , SDV*n*, , , SDV*n*

Abaqus/CAE Usage: Defining initial solution-dependent state variables by importing field data from an output database (`.sim`) file is not supported in Abaqus/CAE.

Defining the Initial Values of Solution-Dependent State Variables for Rebars

The initial values of solution-dependent variables can also be defined for rebars within elements. Rebars are discussed in *Defining Rebar as an Element Property*.

Input File Usage: **INITIAL CONDITIONS*, TYPE=SOLUTION, REBAR

Abaqus/CAE Usage: Initial solution-dependent state variables are not supported in Abaqus/CAE.

Defining the Initial Values of Solution-Dependent State Variables in User Subroutine SDVINI

For complicated cases in Abaqus/Standard user subroutine *SDVINI* can be used to define the initial values of solution-dependent state variables. In this case Abaqus/Standardmakes a call to subroutine *SDVINI* at the start of the analysis for each material integration point in the model. You can then define all solution-dependent state variables at each point as functions of coordinates, element number, etc.

Input File Usage: **INITIAL CONDITIONS*, TYPE=SOLUTION, USER

Abaqus/CAE Usage: User subroutine *SDVINI* is not supported in Abaqus/CAE.

Defining Initial Specific Energy for Equations of State

In Abaqus/Explicit you can specify the initial values of the specific energy for equations of state (see *Equation of State*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=SPECIFIC ENERGY

Abaqus/CAE Usage: Initial specific energy is not supported in Abaqus/CAE.

Defining Spud Can Embedment or Spud Can Preload

In Abaqus/Standard you can define an initial embedment of a spud can. Alternatively, you can define an initial vertical preload of a spud can (see *Elastic-Plastic Joints*).

Input File Usage: Use one of the following options:

**INITIAL CONDITIONS*, TYPE=SPUD EMBEDMENT
**INITIAL CONDITIONS*, TYPE=SPUD PRELOAD

Abaqus/CAE Usage: Initial spud can embedment and preload are not supported in Abaqus/CAE.

Defining Initial Stresses

You can define an initial stress field. Initial stresses can be defined directly or, in Abaqus/Standard, by user subroutine *SIGINI*. Stress values given directly are applied uniformly over the element unless they are defined at each section point through the thickness in shell elements.

If a local coordinate system was defined (see *Orientations*), stresses must be given in the local system.

In soils (porous medium) problems the initial effective stress should be given; see *Coupled Pore Fluid Diffusion and Stress Analysis* for a discussion of defining initial conditions in porous media. The initial effective stress conditions defined for an element are assumed to act on the initial configuration of the element. If the initial

effective stress conditions are removed during the step, the element returns to a stress-free configuration that is different from the initial one. Since displacements and total strain output are measured relative to the initial configuration, the stress-free configuration will have nonzero values for the displacement and total strain fields that will depend on the initial conditions. While it is easy to verify the above behavior analytically in a one-element problem subjected to an initial stress field, the situation in a complex boundary value problem is determined by other factors that may make it difficult to resolve analytically.

If the section properties of beam elements or shell elements are defined by a general section, the initial stress values are applied as initial section forces and moments. In the case of beams initial conditions can be specified only for the axial force, the bending moments, and the twisting moment. In the case of shells initial conditions can be specified only for the membrane forces, the bending moments, and the twisting moment. In both shells and beams initial conditions cannot be prescribed for the transverse shear forces.

Initial contact stresses are calculated internally (see *Initial Contact Stresses in Abaqus/Standard*) by default providing initial conditions for contact. These initial contact stress computations use the initial stresses that you specify in the underlying elements of the contact surfaces and are meant to provide an approximate equilibrating contact traction at contact interfaces.

Initial stress fields cannot be defined for spring elements. See *Springs* for a discussion of defining initial forces in spring elements.

Initial stress fields cannot be defined for elements using a fabric material. However, an initial stress and strain state can be introduced in a fabric material made of membrane elements by defining a reference mesh (see *Defining a Reference Mesh for Membrane Elements and Three-Dimensional Solid Elements* above).

Input File Usage: **INITIAL CONDITIONS*, TYPE=STRESS

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Mechanical** for the **Category** and **Stress** for the **Types for Selected Step**

Defining Initial Stresses for Rebars

Initial values of stress can also be defined for rebars within elements (see *Defining Rebar as an Element Property*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=STRESS, REBAR

Abaqus/CAE Usage: Initial stress for rebars is not supported in Abaqus/CAE.

Defining Initial Stresses That Vary through the Thickness of Shell Elements

Initial values of stress can be defined at each section point through the thickness of shell elements.

Input File Usage: **INITIAL CONDITIONS*, TYPE=STRESS, SECTION POINTS

Abaqus/CAE Usage: Defining initial stress that varies through the thickness of shell elements is not supported in Abaqus/CAE.

Defining Initial Stresses in User Subroutine `SIGINI`

For complicated cases (such as elbow elements) in Abaqus/Standard the initial stress field can be defined by user subroutine *SIGINI*. In this case Abaqus/Standard makes a call to subroutine *SIGINI* at the start of the analysis

for each material calculation point in the model. You can then define all active stress components at each point as functions of coordinates, element number, etc.

Input File Usage: **INITIAL CONDITIONS*, TYPE=STRESS, USER

Abaqus/CAE Usage: User subroutine *SIGINI* is not supported in Abaqus/CAE.

Defining Initial Stresses Using Stress Output from a User-Specified Output Database File

You can define initial stresses using stress output variables from a particular step and increment in the output database (`.odb` or `.sim`) file of a previous Abaqus/Standard analysis. This option is available only for continuum elements when the stress output in the previous analysis was requested at the integration points or at the centroid of the element.

The part (`.prt`) file from the previous analysis is required to read the initial values of predefined stresses from the results file (*Assembly Definition*). In this case both the previous model and the current model must be defined consistently. The element numbering and element types must be the same in both models. If the models are defined in terms of an assembly of part instances, part instance naming must be the same.

The file extension is optional; however, only the output database (`.odb` or `.sim`) file can be used. If no extension is specified, the `.odb` file is used.

Input File Usage: **INITIAL CONDITIONS*, TYPE=STRESS, FILE=*file*, STEP=*step*, INC=*inc*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Mechanical** for the **Category** and **Stress** for the **Types for Selected Step**; select region; **Specification: From output database file**

Defining Initial Stresses by Importing Field Data from an Output Database File

For three-dimensional continuum elements, you can define initial stresses by importing field data from a particular step and increment or a user-specified time in the output database (`.sim`) file of a previous analysis. For more information, see *Importing Data from an Output Database File*.

Input File Usage: **INITIAL CONDITIONS*, TYPE=STRESS
**EXTERNAL FIELD*, FILE=*file name*

Abaqus/CAE Usage: Defining initial stresses by importing field data from an output database (`.sim`) file is not supported in Abaqus/CAE.

Defining Initial Stresses in the Fully Relaxed State

By default, Abaqus assumes that the initial stresses for a viscoelastic material are instantaneous. However, often the initial stresses that are known correspond to a fully relaxed state, which is achieved after a material is kept at a constant strain for a long period of time. Although the initial stresses are identical in both cases, the initial material states and their subsequent responses are different.

Prescribing initial stresses in the fully relaxed state is supported only with finite-strain linear viscoelasticity. In addition, it is not supported for beam and truss elements in Abaqus/Explicit.

Input File Usage: **INITIAL CONDITIONS*, TYPE=STRESS, LONG TERM

Abaqus/CAE Usage: Defining initial stresses in the fully relaxed state is not supported in Abaqus/CAE.

Defining Initial Stresses for Materials with the Mullins Effect Model

When initial stresses are specified for a material with Mullins effect, Abaqus initializes the state assuming that the material followed a monotonic loading path up to those initial stress values. For example, if an initial stress S_0 is specified, Abaqus computes an initial state that corresponds to point A (S_0, λ_1), which is on the loading path (see *Figure 1*).

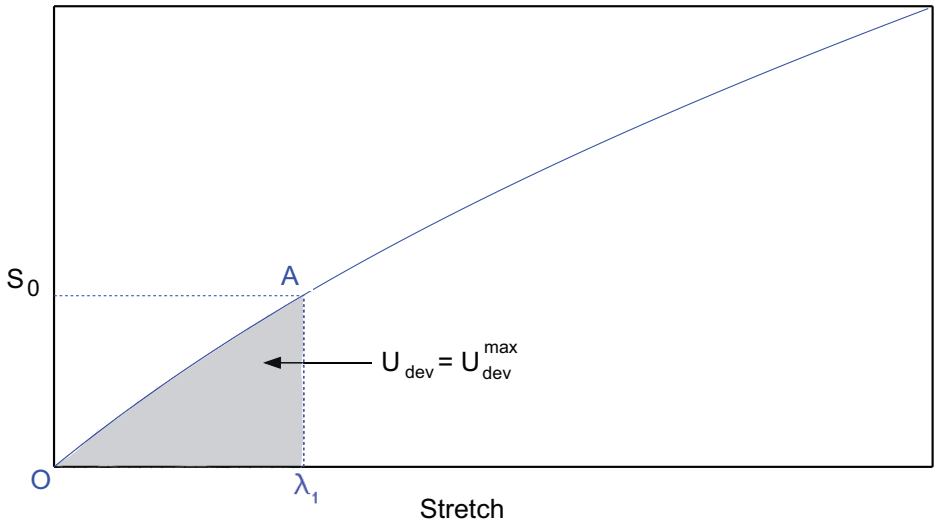


Figure 1: Initial stress for a material with the Mullins effect model specified on the loading path

However, if the material has a non-monotonic deformation history that includes unloading from point B, the initial state corresponding to the initial stress should be point D (S_0, λ_2) (see *Figure 2*).

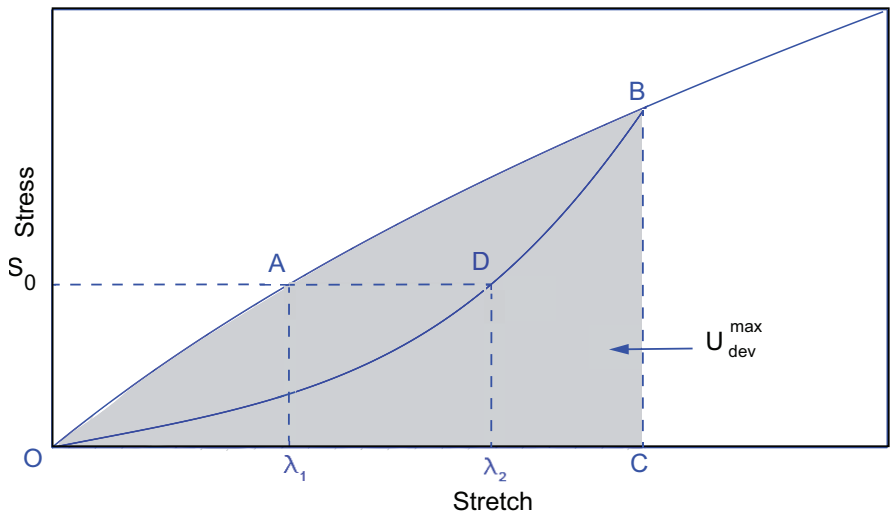


Figure 2: Initial stress for a material with a Mullins effect model specified on the unloading path

In this situation, in addition to specifying initial stresses, you also need to specify the initial maximum energy density, U_{dev}^{max}, experienced by the material, so that Abaqus can compute the correct state accounting for Mullins

effect. If the initial maximum energy density specified is larger than the value computed using the initial stresses, Abaqus computes the state on the unloading path (point D); otherwise, Abaqus ignores the initial maximum energy density and computes the state on the loading path (point A).

Input File Usage: Use the following options to define initial stresses in conjunction with the maximum initial energy density in the Mullins effect model:

**INITIAL CONDITIONS*, TYPE=STRESS

**INITIAL CONDITIONS*, TYPE=MULLINS

Establishing Equilibrium in Abaqus/Standard

When initial stresses are given in Abaqus/Standard (including prestressing in reinforced concrete or interpolation of an old solution onto a new mesh), the initial stress state might not be an exact equilibrium state for the finite element model. Therefore, an initial step should be included to allow Abaqus/Standard to check for equilibrium and iterate, if necessary, to achieve equilibrium.

In a soils analysis (that is, for models containing elements that include pore fluid pressure as a variable) the geostatic stress field procedure (*Geostatic Stress State*) should be used for the equilibrating step. Any initial loading (such as geostatic gravity loads) that contributes to the initial equilibrium should be included in this step definition. The initial time increment and the total time specified in this step should be the same. The initial stresses are applied in full at time zero; and if equilibrium can be achieved, this step converges in one increment. Therefore, there is no benefit to incrementing.

To achieve equilibrium for all other analyses, a first step using the static procedure (*Static Stress Analysis*) should be used. It is recommended that you specify the initial time increment to be equal to the total time specified in this step so that Abaqus/Standard attempts to find equilibrium in one increment.

By default, Abaqus/Standard adopts a ramping technique over the first step. This allows Abaqus/Standard to use automatic incrementation if equilibrium cannot be found in one increment. This ramping is achieved in the following manner:

1. An additional set of artificial or unbalanced stresses is defined at each material point. These stresses are equal in magnitude to the initial stresses but are of opposite sign. Therefore, the sum of the material point initial stresses and these artificial stresses creates zero internal forces at the beginning of the step.
2. The internal unbalanced stresses are ramped off linearly in time during the first step. Therefore, at the end of the step the artificial stresses have been removed completely and the remaining stresses in the material are the stress state in equilibrium.

You can force Abaqus/Standard to achieve equilibrium in one increment by using a step variation on the initial condition to resolve the unbalanced stress instead of ramping the stress down over the entire step. If Abaqus/Standard cannot achieve equilibrium in one increment, the analysis ends.

If the equilibrating step does not converge, it indicates that the initial stress state is so far from equilibrium with the applied loads that significantly large deformations would be generated. This is generally not the intention of an initial stress state; therefore, it suggests that you should recheck the specified initial stresses and loads.

Input File Usage: Use one of the following options to specify how the unbalanced stress should be resolved:

**INITIAL CONDITIONS*, TYPE=STRESS,
UNBALANCED STRESS=RAMP (default)
**INITIAL CONDITIONS*, TYPE=STRESS,
UNBALANCED STRESS=STEP

Abaqus/CAE Usage: Initial equilibrium stress is not supported in Abaqus/CAE.

Establishing Equilibrium in Abaqus/Explicit

Abaqus/Explicit computes the initial acceleration at nodes taking into account the initial stresses, the loads, and the boundary conditions in the initial configuration. For an initially static problem, the specified boundary conditions, the initial stresses, and the initial loading should be consistent with a static equilibrium. Otherwise, the solution is likely to be noisy. The noise might be reduced by introducing a dummy step with a temporary viscous loading to attempt to reestablish a static equilibrium. Alternatively, you can introduce an initial short step in which all degrees of freedom are fixed with boundary conditions (all initial loads should be included in this initial step); in a second step, release all but the actual boundary conditions.

Defining Elevation-Dependent (Geostatic) Initial Stresses

You can define elevation-dependent initial stresses. When a geostatic stress state is prescribed for a particular element set, the stress in the vertical direction (assumed to be the z-direction in three-dimensional and axisymmetric models and the y-direction in two-dimensional models) is assumed to vary (piecewise) linearly with this vertical coordinate.

For the vertical stress component, you must give two pairs of stress and elevation values to define the stress throughout the element set. For material points lying between the two elevations given, Abaqus will use linear interpolation to determine the initial stress; for points lying outside the two elevations given, Abaqus will use linear extrapolation. In addition, horizontal (lateral) stress components are given by entering one or two "coefficients of lateral stress," which define the lateral direct stress components as the vertical stress at the point multiplied by the value of the coefficient. In axisymmetric cases only one value of the coefficient of lateral stress is used and, therefore, only one value need be entered.

Geostatic initial stresses are for use with continuum elements only. In Abaqus/Standard elevation-dependent initial stresses should be specified for beams and shells in user subroutine *SIGINI*, as explained earlier. In Abaqus/Explicit elevation-dependent initial stresses cannot be specified for beams and shells.

The geostatic stress state specified initially should be in equilibrium with the applied loads (such as gravity) and boundary conditions. An initial step should be included to allow Abaqus to check for equilibrium after this interpolation has been done; see the discussion above on establishing equilibrium when an initial stress field is applied.

Input File Usage: **INITIAL CONDITIONS*, TYPE=STRESS, GEOSTATIC

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Mechanical** for the **Category** and **Geostatic stress** for the **Types for Selected Step**

Defining Initial Temperatures

You can define initial temperatures at the nodes of either heat transfer or stress/displacement elements. The temperatures of stress/displacement elements can be changed during an analysis (see *Predefined Fields*).

The definition of initial temperature values must be compatible with the section definition of the element and with adjacent elements, as explained in *Predefined Fields*.

Input File Usage: **INITIAL CONDITIONS*, TYPE=TEMPERATURE

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**

Defining Uniform Initial Temperatures

You can apply uniform initial temperatures to either the entire model or to node sets that you specify. Omit the node number or node set to apply the specified temperature to all nodes in the model automatically.

You can specify uniform temperature with all element types, including beams and shells. The specified uniform temperature is applied to all section points in beams and shells. However, the definition of initial temperature values must be compatible with the section definition of the element and with adjacent elements, as explained in *Predefined Fields*. Abaqus issues a warning message during input file preprocessing if an initial temperature is applied to any node that is associated with at least one shell or beam section that specifies temperatures using gradients and at least one section that directly specifies the values of temperature.

Input File Usage: **INITIAL CONDITIONS*, TYPE=TEMPERATURE, SECTION SPECIFICATION=UNIFORM

Abaqus/CAE Usage: Defining uniform initial temperatures is not supported in Abaqus/CAE.

Defining Initial Temperatures from a User-Specified Results or Output Database File

You can define initial temperatures as those values existing as nodal temperatures at a particular step and increment in the results or output database file of a previous Abaqus/Standard heat transfer analysis (see *Predefined Fields*).

The part (`.prt`) file from the previous analysis is required to read initial temperatures from the results or output database file (see *Assembly Definition*). Both the previous model and the current model must be consistently defined in terms of an assembly of part instances; node numbering must be the same, and part instance naming must be the same.

The file extension is optional; however, if both results and output database files exist, the results file will be used.

Input File Usage: **INITIAL CONDITIONS*, TYPE=TEMPERATURE, FILE=*file*, STEP=*step*, INC=*inc*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**: select region: **Distribution: From results or output database file**, **File name:** *file*, **Step:** *step*, and **Increment:** *inc*

Defining Initial Temperatures by Importing Field Data from an Output Database File

For three-dimensional continuum elements, you can define initial temperatures by importing field data as nodal temperatures at a particular step and increment or a user-specified time in the output database (`.sim`) file of a previous analysis. For more information, see *Importing Data from an Output Database File*.

Input File Usage: **INITIAL CONDITIONS*, TYPE=TEMPERATURE
**EXTERNAL FIELD*, FILE=*file name*

Abaqus/CAE Usage: Defining initial temperatures by importing field data from an output database (`.sim`) file is not supported in Abaqus/CAE.

Interpolating Initial Temperatures for Dissimilar Meshes from a User-Specified Results or Output Database File

When the mesh for the heat transfer analysis is different from the mesh for the subsequent stress/displacement analysis, Abaqus can interpolate the temperature values from the nodes in the undeformed heat transfer model to the current nodal temperatures. This technique can also be used in cases where the meshes match but the node number or part instance naming differs between the analyses. Only temperatures from an output database file can be used for the interpolation; Abaqus will look for the `.odb` extension automatically. The part (`.prt`) file from the previous analysis is required if that analysis model is defined in terms of an assembly of part instances (see *Assembly Definition*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=TEMPERATURE, INTERPOLATE, FILE=*file*, STEP=*step*, INC=*inc*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**: select region: **Distribution: From results or output database file**, **File name:** *file*, **Mesh compatibility: Incompatible**

Interpolating Initial Temperatures for Dissimilar Meshes with User-Specified Regions

When regions of elements in the heat transfer analysis are close or touching, the dissimilar mesh interpolation capability can result in an ambiguous temperature association. For example, consider a node in the current model that lies on or close to a boundary between two adjacent parts in the heat transfer model, and consider a case where temperatures in these parts are different. When interpolating, Abaqus will identify a corresponding parent element at the boundary for this node from the heat transfer analysis. This parent element identification is done using a tolerance-based search method. Hence, in this example the parent element might be found in either of the adjacent parts, resulting in an ambiguous temperature definition at the node. You can eliminate this ambiguity by specifying the source regions from which temperatures are to be interpolated. The source region refers to the heat transfer analysis and is specified by an element set. The target region refers to the current analysis and is specified by a node set.

Input File Usage: **INITIAL CONDITIONS*, TYPE=TEMPERATURE, INTERPOLATE, FILE=*file*, STEP=*step*, INC=*inc*, DRIVING ELSETS

Abaqus/CAE Usage: You cannot specify the regions where temperatures are to be interpolated in Abaqus/CAE.

Interpolating Initial Temperatures for Meshes That Differ Only in Element Order from a User-Specified Results or Output Database File

If the only difference in the meshes is the element order (first-order elements in the heat transfer model and second-order elements in the stress/displacement model), in Abaqus/Standard you can indicate that midside node temperatures in second-order elements are to be interpolated from corner node temperatures read from the results or output database file of the previous heat transfer analysis using first-order elements. You must ensure that the corner node temperatures are not defined using a mixture of direct data input and reading from the results or output database file, since midside node temperatures that give unrealistic temperature fields might result. In practice, the capability for calculating midside node temperatures is most useful when temperatures generated

by a heat transfer analysis are read from the results or output database file for the whole mesh during the stress analysis. Once the midside node capability is activated, the capability remains active for the rest of the analysis, including for any predefined temperature fields defined to change temperatures during the analysis. The general interpolation and midside node capabilities are mutually exclusive.

Input File Usage: **INITIAL CONDITIONS*, TYPE=TEMPERATURE, MIDSIDE, FILE=*file*, STEP=*step*, INC=*inc*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**: select region: **Distribution: From results or output database file**, **File name:** *file*, **Step:** *step*, **Increment:** *inc*, **Mesh compatibility: Compatible**, and toggle on **Interpolate midside nodes**

Defining the Initial Configuration in a One-Step Inverse Analysis

In a one-step inverse analysis you must define the initial configuration by specifying initial conditions for the unfolded coordinates of all nodes in the part. One-step inverse analysis uses the Newton method, which requires an initial estimate of the solution as a starting point in the iterative algorithm. This starting point is defined by specifying unfold coordinate initial conditions.

Input File Usage: **INITIAL CONDITIONS*, TYPE=UNFOLD COORDINATE

Abaqus/CAE Usage: One-step inverse analysis is not supported in Abaqus/CAE.

Defining Initial Velocities for Specified Degrees of Freedom

You can define initial velocities for specified degrees of freedom. When initial velocities are given for dynamic analysis, they should be consistent with all the constraints on the model, especially time-dependent boundary conditions. Abaqus ensures that they are consistent with boundary conditions and with multi-point and equation constraints but does not check for consistency with internal constraints such as incompressibility of the material. In case of conflict, boundary conditions take precedence over initial conditions.

Initial velocities must be defined in global directions, regardless of the use of local transformations (*Transformed Coordinate Systems*).

Input File Usage: **INITIAL CONDITIONS*, TYPE=VELOCITY

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Mechanical** for the **Category** and **Velocity** for the **Types for Selected Step**

Defining Initial Volume Fractions for Eulerian Elements

You can define initial volume fractions to create material within Eulerian elements in Abaqus/Explicit. By default, these elements are filled with void. See *Initial Conditions* for a description of strategies for initializing Eulerian materials.

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: choose **Other** for the **Category** and **Material Assignment** for the **Types for Selected Step**

Boundary Conditions

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *Abaqus Model Definition*
- *About Prescribed Conditions*
- *VDISP*
- *DISP*
- **BOUNDARY*
- *Using the boundary condition editors*

Overview

You can prescribe values of basic solution variables, including displacements and rotations in stress/displacement analysis and temperature in heat transfer or coupled thermal-stress analysis.

Boundary conditions:

- can be used to specify the values of all basic solution variables (displacements, rotations, warping amplitude, fluid pressures, pore pressures, temperatures, electrical potentials, normalized concentrations, acoustic pressures, or connector material flow) at nodes;
- can be given as "model" input data (within the initial step in Abaqus/CAE) to define zero-valued boundary conditions;
- can be given as "history" input data (within an analysis step) to add, modify, or remove zero-valued or nonzero boundary conditions; and
- can be defined by the user through subroutines *DISP* for Abaqus/Standard and *VDISP* for Abaqus/Explicit.

Relative motions in connector elements can be prescribed similar to boundary conditions. See *Connector Actuation* for more detailed information.

Prescribing Boundary Conditions as Model Data

Only zero-valued boundary conditions can be prescribed as model data (i.e., in the initial step in Abaqus/CAE). You can specify the data using either "direct" or "type" format. As described below, the "type" format is a way of conveniently specifying common types of boundary conditions in stress/displacement analyses. "Direct" format must be used in all other analysis types.

For both "direct" and "type" format you specify the region of the model to which the boundary conditions apply and the degrees of freedom to be restrained. (See *Conventions* for the degree of freedom numbers used in Abaqus.)

Boundary conditions prescribed as model data can be modified or removed during analysis steps.

Input File Usage: **BOUNDARY*

Any number of data lines can be used to specify boundary conditions, and in stress/displacement analyses both "direct" and "type" format can be specified with a single use of the **BOUNDARY* option.

Abaqus/CAE Usage: Load module: **Create Boundary Condition**: **Step: Initial**

Using the Direct Format

You can choose to enter the degrees of freedom to be constrained directly.

Input File Usage: Either a single degree of freedom or the first and last of a range of degrees of freedom can be specified.

**BOUNDARY*
node or node set, *degree of freedom*
**BOUNDARY*
node or node set, *first degree of freedom*, *last degree of freedom*

For example,

**BOUNDARY*
EDGE, 1

indicates that all nodes in node set `EDGE` are constrained in degree of freedom 1 (u_x), while the data line

EDGE, 1, 4

indicates that all nodes in node set `EDGE` are constrained in degrees of freedom 1–4 (u_x, u_y, u_z, ϕ_x).

Abaqus/CAE Usage: Load module: **Create Boundary Condition**: **Step: Initial**

Use one of the following options:

Category: Mechanical; **Displacement/Rotation**, **Velocity/Angular velocity**, or **Acceleration/Angular acceleration**; select regions and toggle on the degree or degrees of freedom

Category: Electrical/Magnetic; **Electric potential**; select regions

Category: Other; **Temperature**, **Pore pressure**, **Mass concentration**, **Acoustic pressure**, or **Connector material flow**; select regions

If you are specifying a temperature boundary condition for a shell region, you can enter multiple degrees of freedom, from 11 to 31, inclusive.

Using the "Type" Format in Stress/Displacement Analyses

The type of boundary condition can be specified instead of degrees of freedom. The following boundary condition "types" are available in both Abaqus/Standard and Abaqus/Explicit:

XSYMM

Symmetry about a plane $X = \text{constant}$ (degrees of freedom $1, 5, 6 = 0$).

YSYMM

Symmetry about a plane $Y = \text{constant}$ (degrees of freedom $2, 4, 6 = 0$).

ZSYMM

Symmetry about a plane $Z = \text{constant}$ (degrees of freedom $3, 4, 5 = 0$).

ENCASTRE

Fully built-in (degrees of freedom $1, 2, 3, 4, 5, 6 = 0$).

PINNED

Pinned (degrees of freedom $1, 2, 3 = 0$).

The following boundary condition types are available only in Abaqus/Standard:

XASYMM

Antisymmetry about a plane with $X = \text{constant}$ (degrees of freedom 2, 3, 4 $= 0$).

YASYMM

Antisymmetry about a plane with $Y = \text{constant}$ (degrees of freedom 1, 3, 5 $= 0$).

ZASYMM

Antisymmetry about a plane with $Z = \text{constant}$ (degrees of freedom 1, 2, 6 $= 0$).

Warning: When boundary conditions are prescribed at a node in an analysis involving finite rotations, at least two rotation degrees of freedom should be constrained. Otherwise, the prescribed rotation at the node may not be what you expect. Therefore, antisymmetry boundary conditions should generally not be used in problems involving finite rotations.

NOWARP

Prevent warping of an elbow section at a node.

NOOVAL

Prevent ovalization of an elbow section at a node.

NODEFORM

Prevent all cross-sectional deformation (warping, ovalization, and uniform radial expansion) at a node.

The NOWARP, NOOVAL, and NODEFORM types apply only to elbow elements (*Pipes and Pipebends with Deforming Cross-Sections: Elbow Elements*).

For example, applying a boundary condition of type XSYMM to node set `EDGE` indicates that the node set lies on a plane of symmetry that is normal to the *X*-axis (which will be the global *X*-axis or the local *X*-axis if a nodal transformation has been applied at these nodes). This boundary condition is identical to applying a boundary condition using the direct format to degrees of freedom 1, 5, and 6 in node set `EDGE` since symmetry about a plane *X*=constant implies $u_x = 0$, $\phi_y = 0$, and $\phi_z = 0$.

Once a degree of freedom has been constrained using a "type" boundary condition as model data, the constraint cannot be modified by using a boundary condition in "direct" format as model data; modifying a constraint in such a way will only produce an error message in the data (`.dat`) file indicating that conflicting boundary conditions exist in the model data.

Input File Usage: **BOUNDARY*
node or node set, *boundary condition type*

Abaqus/CAE Usage: Load module: **Create Boundary Condition**: **Step: Initial**: **Symmetry/Antisymmetry/Encastre**: select regions and toggle on the boundary condition type

Prescribing Boundary Conditions at Phantom Nodes for Enriched Elements

Phantom nodes for an enriched element can be either colocated with real nodes or located on an element edge between two real corner nodes (see *Modeling Discontinuities as an Enriched Feature Using the Extended Finite Element Method*). For phantom nodes coincident with real nodes, boundary conditions can be specified using the node numbers of the real nodes.

Alternatively, for phantom nodes with pore pressure degrees of freedom that are located on an element edge, you can specify the boundary conditions by identifying the phantom nodes in terms of the two real corner node numbers or by indicating they will be interpolated from the specified real corner nodes when the enriched element is cracked.

Input File Usage: Use the following option to specify boundary conditions at a phantom node originally located coincident with the specified real node:

**BOUNDARY*, PHANTOM=NODE
node number, *first degree of freedom*, *last degree of freedom*

Use the following option to specify boundary conditions at a phantom node located at an element edge:

**BOUNDARY*, PHANTOM=EDGE
first corner node number, *second corner node number*, *first degree of freedom*, *last degree of freedom*

Use the following option to indicate that the boundary conditions applied to a phantom node located at an element edge will be interpolated automatically from the specified real corner nodes when the enriched element is cracked:

*BOUNDARY, PHANTOM=INCLUDED
node or node set, *first degree of freedom*, *last degree of freedom*

Abaqus/CAE Usage: Prescribing boundary conditions at phantom nodes for enriched elements is not supported in Abaqus/CAE.

Prescribing Boundary Conditions as History Data

Boundary conditions can be prescribed within an analysis step using either "direct" or "type" format. As with model data boundary conditions, the "type" format can be used only in stress/displacement analyses; whereas, the "direct" format can be used in analysis types.

When using the "direct" format, boundary conditions can be defined as the total value of a variable or, in a stress/displacement analysis, as the value of a variable's velocity or acceleration.

As many boundary conditions as necessary can be defined in a step.

Input File Usage: **BOUNDARY*

Abaqus/CAE Usage: Load module: **Create Boundary Condition**: **Step:** ***analysis_step***

Using the Direct Format

Specify the region of the model to which the boundary conditions apply, the degree or degrees of freedom to be specified (see *Conventions* for the degree of freedom numbers used in Abaqus), and the magnitude of the boundary condition. If the magnitude is omitted, it is the same as specifying a zero magnitude.

In stress/displacement analysis you can specify a velocity history or an acceleration history. The default is a displacement history.

Input File Usage: Use either of the following options to prescribe a displacement history:

**BOUNDARY* or **BOUNDARY*, TYPE=DISPLACEMENT
node or node set, *degree of freedom*, , *magnitude*
node or node set, *first degree of freedom*, *last degree of freedom*, *magnitude*

Use the following option to prescribe a velocity history (the data lines are the same as above):

**BOUNDARY*, TYPE=VELOCITY

Use the following option to prescribe an acceleration history (the data lines are the same as above):

**BOUNDARY*, TYPE=ACCELERATION

For example,

**BOUNDARY*, TYPE=VELOCITY
EDGE, 1, 1, 0.5

indicates that all nodes in node set EDGE have a prescribed velocity magnitude of 0.5 in degree of freedom 1 (u_x).

Abaqus/CAE Usage: Load module: **Create Boundary Condition**: **Step:** ***analysis_step***:

Select one of the following categories and types:

Category: Mechanical; **Displacement/Rotation**; select regions; **Distribution: Uniform** or select an analytical field or a discrete field; toggle on the degree or degrees of freedom; *magnitude*

Category: Mechanical; **Velocity/Angular velocity** or **Acceleration/Angular acceleration**; select regions; **Distribution: Uniform** or select an analytical field; toggle on the degree or degrees of freedom; *magnitude*

Category: Electrical/Magnetic; **Electric potential**; select regions; **Distribution: Uniform** or select an analytical field; **Method: Specify magnitude**; *magnitude*

Category: Other; **Temperature**, **Pore pressure**, **Mass concentration**, **Acoustic pressure**, or **Connector material flow**; select regions; **Distribution: Uniform** or select an analytical field; **Method: Specify magnitude**; *magnitude*

If you are specifying a temperature boundary condition for a shell region, you can enter multiple degrees of freedom, from 11 to 31, inclusive.

Prescribed Displacement

In Abaqus/Standard you can prescribe jumps in displacements. For example, a displacement-type boundary condition is used to apply a prescribed displacement magnitude of 0.5 in degree of freedom 1 (u_x) to the nodes in node set EDGE. In a second step these nodes can be moved by another 0.5 length units (to a total displacement of 1.0) by applying a prescribed displacement magnitude of 1.0 in degree of freedom 1 to node set EDGE. Specifying a prescribed displacement magnitude of 0 (or omitting the magnitude) in degree of freedom 1 in the next step would return the nodes in node set EDGE to their original locations.

In contrast, Abaqus/Explicit does not admit jumps in displacements and rotations. Displacement boundary conditions in displacement and rotation degrees of freedom are enforced in an incremental manner using the slope of the amplitude curve (see below). If no amplitude is specified, Abaqus/Explicit will ignore the user-supplied displacement value and enforce a zero velocity boundary condition.

The displacement must remain continuous across steps. If amplitude curves are specified, it is possible, but not valid, to specify a jump in the displacement across a step boundary when using step time for the amplitude definition. Abaqus/Explicit will ignore such jumps in displacement if they are specified.

Using the “Type” Format in Stress/Displacement Analyses

The type of boundary condition can be specified (as history data) instead of degrees of freedom in the same manner as discussed above for model data. The boundary condition “types” that are available as history data are the same as those available as model data.

Once a degree of freedom has been constrained using a “type” boundary condition as history data, the constraint cannot be modified by using a boundary condition in “direct” format. The constraint can be redefined only by using a boundary condition in “direct” format after all previously applied boundary conditions specified using “type” format are removed.

Input File Usage: **BOUNDARY*
node or node set, *boundary condition type*

Abaqus/CAE Usage: Load module: **Create Boundary Condition**: **Step:** ***analysis_step***: **Symmetry/Antisymmetry/Encastre**: select regions and toggle on the boundary condition type

Prescribing Boundary Conditions at Phantom Nodes for Enriched Elements

You can specify boundary conditions at phantom nodes as history data in the same manner as discussed above for model data (see *Modeling Discontinuities as an Enriched Feature Using the Extended Finite Element Method* for more information on enriched elements). To specify nonzero boundary conditions, enter the actual magnitude.

Input File Usage: Use the following option to specify boundary conditions at a phantom node originally located coincident with the specified real node:

**BOUNDARY*, PHANTOM=NODE
node number, *first degree of freedom*, *last degree of freedom*, *magnitude*

Use the following option to specify boundary conditions at a phantom node located at an element edge:

**BOUNDARY*, PHANTOM=EDGE
first corner node number, *second corner node number*, *first degree of freedom*, *last degree of freedom*, *magnitude*

Use the following option to indicate that the boundary conditions applied to a phantom node located at an element edge will be interpolated automatically from the specified real corner nodes when the enriched element is cracked:

**BOUNDARY*, PHANTOM=INCLUDED
node or node set, *first degree of freedom*, *last degree of freedom*, *magnitude*

Abaqus/CAE Usage: Prescribing boundary conditions at phantom nodes for enriched elements is not supported in Abaqus/CAE.

Defining Boundary Conditions That Vary with Time

The prescribed magnitude of a basic solution variable, a velocity, or an acceleration can vary with time during a step according to an amplitude definition (*Amplitude Curves*).

When an amplitude definition is used with a boundary condition in a dynamic or modal dynamic analysis, the first and second time derivatives of the constrained variable may be discontinuous. For example, Abaqus will compute the corresponding velocity and acceleration from a given displacement boundary condition.

By default, Abaqus/Standard will smooth the amplitude curve so that the derivatives of the specified boundary condition will be finite. You must ensure that the applied values are correct after smoothing.

Abaqus/Explicit does not apply default smoothing to discontinuous amplitude curves. To avoid the “noisy” solution that may result from discontinuities in Abaqus/Explicit, it is better to specify the velocity history of a node. See *Amplitude Curves*.

Input File Usage: Use both of the following options:

**AMPLITUDE*, NAME=*name*
**BOUNDARY*, AMPLITUDE=*name*

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **Name**: *amplitude_name*
Load module: **Create Boundary Condition**: **Step:** ***analysis_step***: ***boundary condition***; **Amplitude**: *amplitude_name*

Defining Boundary Conditions through User Subroutines

If an amplitude based evolution of a boundary condition is not sufficient, you can define it yourself in a user subroutine. For this purpose, Abaqus/Standard provides the routine *DISP*; whereas, Abaqus/Explicit provides the routine *VDISP*. The region to which the boundary conditions apply and the constrained degrees of freedom are specified as part of the boundary condition definition. The actual boundary condition is set within the user routine based on a number of variables made available in those routines (see *DISP* and *VDISP*).

Abaqus/Standard allows for an amplitude and a reference magnitude definition for a user-defined boundary condition and you may overwrite the amplitude based boundary value within the *DISP* routine. Whereas, Abaqus/Explicit ignores the reference magnitude, but passes in the amplitude value as an argument to the user routine *VDISP* and you may define the boundary condition to a nonzero value.

Input File Usage: **BOUNDARY*, USER

Abaqus/CAE Usage: Load module: **Create Boundary Condition**: **Step:** ***analysis_step***; ***boundary condition***; **Distribution: User-defined**

Boundary Condition Propagation

By default, all boundary conditions defined in the previous general analysis step remain unchanged in the subsequent general step or in subsequent consecutive linear perturbation steps. Boundary conditions do not propagate between linear perturbation steps.

You define the boundary conditions in effect for a given step relative to the preexisting boundary conditions. At each new step the existing boundary conditions can be modified and additional boundary conditions can be specified. Alternatively, you can release all previously applied boundary conditions in a step and specify new ones. In this case any boundary conditions that are to be retained must be respecified.

If you use a step time amplitude curve when respecifying a boundary condition that is to be retained from a previous step, the prescribed magnitude of the variable will vary according to the amplitude definition.

Modifying Boundary Conditions

When you modify an existing boundary condition, the node or node set must be specified in exactly the same way as previously. For example, if a boundary condition is specified for a node set in one step and for an individual node contained in the set in another step, Abaqus issues an error. You must remove the boundary condition and respecify it to change the way the node or node set is specified.

Input File Usage: Use either of the following options to modify an existing boundary condition or to specify an additional boundary condition:

**BOUNDARY*
**BOUNDARY*, OP=MOD

Abaqus/CAE Usage: Load module: **Create Boundary Condition** or **Boundary Condition Manager**: **Edit**

Removing Boundary Conditions

If you choose to remove any boundary condition in a step, no boundary conditions will be propagated from the previous general step. Therefore, all boundary conditions that are in effect during this step must be respecified. The only exception to this rule is during an eigenvalue buckling prediction procedure, as described in *Eigenvalue Buckling Prediction*.

Setting a boundary condition to zero is not the same as removing it.

Input File Usage: Use the following option to release all previously applied boundary conditions and to specify new boundary conditions:

**BOUNDARY*, OP=NEW

If the OP=NEW parameter is used on any **BOUNDARY* option within a step, it must be used on all **BOUNDARY* options in the step.

Abaqus/CAE Usage: Use the following option to remove a boundary condition within a step:

Load module: **Boundary Condition Manager**: **Deactivate**

Abaqus/CAE automatically respecifies any boundary conditions (including step time amplitude curves used in previous steps) that should remain in effect during this step.

Fixing Degrees of Freedom at a Point in an Abaqus/Standard Analysis

In Abaqus/Standard you can “freeze” specified degrees of freedom at their final values from the last general analysis step. Specifying a zero velocity or zero acceleration boundary condition has the same effect as fixing the degrees of freedom for displacement or velocity, respectively. In a one-step inverse procedure, all degrees of freedom are constrained based on the specified initial configuration for the point.

Input File Usage: **BOUNDARY*, FIXED

The OP=NEW parameter must be used with the FIXED parameter if there are any other **BOUNDARY* options in the same step that have the OP=NEW parameter. Any magnitudes given for the boundary condition are ignored. Except for in a one-step inverse procedure, the FIXED parameter is ignored if it is used in the first step of an analysis.

Abaqus/CAE Usage: Load module; **Create Boundary Condition**; **Step:** ***analysis_step***; ***boundary condition***; **Method**: **Fixed at Current Position** (available only if a previous general analysis step exists)

One-step inverse analysis is not supported in Abaqus/CAE.

Prescribing Boundary Conditions in Linear Perturbation Steps

In a linear perturbation step (*General and Perturbation Procedures*) the magnitudes of prescribed boundary conditions should be given as the magnitudes of the perturbations about the base state. Boundary conditions given within the model definition are always regarded as part of the base state, even if the first analysis step is a linear perturbation step. The boundary conditions given in a linear perturbation step will not affect subsequent steps.

If a perturbation step does not contain a boundary condition definition, degrees of freedom that are restrained/prescribed in the base state will be restrained in the perturbation step and will have perturbation magnitudes of zero. To prescribe nonzero perturbation magnitudes, you have to modify the existing boundary conditions. You can also fix and prescribe perturbation magnitudes of degrees of freedom that are unrestrained in the base state.

If degrees of freedom that are restrained/prescribed in the base state are released, all restraints that are to remain must be respecified, remembering that all magnitudes will be interpreted as perturbations.

Fixing the degrees of freedom at their final values from the last general analysis step (see previous discussion) has the same effect as modifying the existing boundary conditions to have zero perturbation magnitudes for all specified degrees of freedom.

The antisymmetric buckling modes of a symmetric structure can be found in an eigenvalue buckling prediction analysis by specifying the proper boundary conditions (see *Eigenvalue Buckling Prediction*).

Prescribing Real and Imaginary Values in Boundary Conditions

In steady-state dynamic and matrix generation procedures, a boundary condition can be prescribed using either a real or an imaginary value (see *Direct-Solution Steady-State Dynamic Analysis* and *Generating Matrices as a Linear Analysis Step*). If the real value is prescribed for a degree of freedom (and the imaginary value is not explicitly prescribed), the imaginary value is considered to be zero. Similarly, if the imaginary value is prescribed (and the real value is not explicitly prescribed), the real value is considered to be zero.

Prescribed Motion in Modal Superposition Procedures

In modal superposition procedures (*About Dynamic Analysis Procedures*) prescribed displacements cannot be defined directly using a boundary condition. Instead, the boundary conditions are grouped into bases in a frequency extraction step. Then, the motion of each base is prescribed in the modal superposition step. See *Natural Frequency Extraction* and *Transient Modal Dynamic Analysis* for details on this method.

Input File Usage: **BOUNDARY*, BASE NAME
**BASE MOTION*

Abaqus/CAE Usage: Load module; **Create Boundary Condition**; **Step:** *modal_dynamic_step*, *steady-state_dynamic_step*, or *random_response_step*; **Category: Mechanical**; **Types for Selected Step: Displacement base motion** or **Velocity base motion** or **Acceleration base motion**

Submodeling

When using the submodeling technique, the magnitudes of the boundary conditions in the submodel can be defined by interpolating the values of the prescribed degrees of freedom from the file output results of the global model. See *Node-Based Submodeling* for details.

Prescribing Large Rotations

Sequential finite rotations about different axes of rotation are not additive, which can make direct specification of such rotations challenging. Abaqus applies rotational boundary conditions by compounding incremental rotations (as described in *Compound rotations*). In this process, Abaqus will ignore any jumps in the specified rotation boundary condition, such as when nondefault amplitudes are used to introduce discontinuities in rotation across step boundaries. This treatment of rotation boundary conditions in Abaqus is independent of whether the axis of rotation changes or not during the sequence of rotations. For these and other reasons, it is much simpler to apply finite-rotation boundary conditions by specifying the rotational velocity versus time. For a discussion of the rotation degrees of freedom and a multiple step finite rotation example that demonstrates why velocity-type boundary conditions are preferred for specifying finite-rotation boundary conditions, see *Conventions*.

When velocity-type boundary conditions are used to prescribe rotations, the definition is given in terms of the angular velocity instead of the total rotation. If the angular velocity is associated with a nondefault amplitude, Abaqus calculates the prescribed increment of rotation as the average of the prescribed angular velocities at the beginning and the end of each increment, multiplied by the time increment.

In Abaqus/Explicit displacement-type boundary conditions that refer to an amplitude curve are effectively enforced as velocity boundary conditions using average velocities over time increments as computed by finite differences of values from the amplitude curve. As with prescribed displacements (see *Prescribed Displacement* above), Abaqus/Explicit does not admit jumps in rotations.

Displacement-type boundary conditions in Abaqus/Standard that constrain just one component of rotation can have essentially no effect on the solution because the two unconstrained rotational degrees of freedom can combine to override the constraint.

Example: Using Velocity-Type Boundary Conditions to Prescribe Rotations

For example, if a rotation of 6π about the z-axis is required in a static step, with no rotation about the x- and y-axes, use a step time (specified as part of the static step definition) of 1.0, and define a velocity-type boundary condition to specify zero velocity for degrees of freedom 4 and 5 and a constant angular velocity of 6π for degree of freedom 6. Since the default variation for a velocity-type boundary condition in a static procedure is a step, the velocity will be constant over the step. Alternatively, an amplitude reference could be used to specify the desired variation over the step.

```
*BOUNDARY, TYPE=VELOCITY
NODE, 4
NODE, 5
NODE, 6, 6, 18.84955592
```

If, in the next step, the same node should have an additional rotation of $\pi/2$ radians about the global *x*-axis, use another static step with a step time of 1.0 and again define a velocity-type boundary condition to prescribe zero velocity for degrees of freedom 5 and 6 and a constant angular velocity of $\pi/2$ for degree of freedom 4.

```
*BOUNDARY, TYPE=VELOCITY
NODE, 4, 4, 1.570796327
NODE, 5
NODE, 6
```

Prescribing Radial Motion on an Axisymmetric Model

The radial coordinate for any node in an axisymmetric model must be positive. Therefore, you must make sure that any specified boundary condition does not violate this condition.

Loads

Many types of loading are available, depending on the analysis procedure.

In this section:

- *About Loads*
- *Concentrated Loads*
- *Distributed Loads*
- *Surface Tension Loads*
- *Fluid Pressure Penetration Loads*
- *Thermal Loads*
- *Electromagnetic Loads*
- *Acoustic and Shock Loads*
- *Pore Fluid Flow*

About Loads

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *General and Perturbation Procedures*
- *About Prescribed Conditions*
- *Concentrated Loads*
- *Distributed Loads*
- *Fluid Pressure Penetration Loads*
- *Thermal Loads*
- *Electromagnetic Loads*
- *Acoustic and Shock Loads*
- *Pore Fluid Flow*
- *Creating and modifying prescribed conditions*
- *Using the load editors*

Overview

External loading can be applied in the following forms: concentrated or distributed tractions, concentrated or distributed fluxes, and incident wave loads.

Many types of distributed loads are provided; they depend on the element type and are described in *About the Element Library*. This section discusses general concepts that apply to all types of loading; see *About Prescribed Conditions* for general information that applies to all types of prescribed conditions.

Concentrated and distributed tractions are discussed in *Concentrated Loads* and *Distributed Loads*, respectively. A special type of distributed load, fluid pressure-penetration load, is discussed in *Fluid Pressure Penetration Loads*. Thermal loading (heat flux) is discussed in *Thermal Loads*. Electromagnetic loads are discussed in *Electromagnetic Loads*. Loads due to incident wave fields such as due to sound sources or an underwater explosion are discussed in *Acoustic and Shock Loads*. Pore fluid flow is discussed in *Pore Fluid Flow*. All other load types, which are applicable to only a single type of analysis, are discussed in the appropriate sections in *Analysis Procedures* and *Analysis Solution and Control*.

In some situations, concentrated loads and some commonly used distributed loads (such as pressure applied on a surface) might rotate during a geometrically nonlinear analysis. Such loads are known as follower loads; further details on follower loads can be found in *Follower Loads in Large-Displacement Analysis*, *Specifying Concentrated Follower Forces*, *Follower Surface Loads*, and *Follower Edge and Line Loads*. Follower loads might also lead to an unsymmetric contribution to the stiffness matrix, which is generally referred to as the load stiffness; some issues related to the load stiffness contribution are discussed in *Improving the Rate of Convergence in Large-Displacement Implicit Analysis* and *Improving the Rate of Convergence in Large-Displacement Implicit Analysis*.

Element-Based Versus Surface-Based Distributed Loads

There are two ways of specifying distributed loads in Abaqus: element-based distributed loads and surface-based distributed loads. Element-based distributed loads can be prescribed on element bodies, element surfaces, or element edges. Surface-based distributed loads can be prescribed on geometric surfaces or geometric edges. In

Abaqus/CAE distributed surface and edge loads can be element-based or surface-based, while distributed body loads are prescribed on geometric bodies or element bodies.

Element-Based Loads

Use element-based loads to define distributed loads on element surfaces, element edges, and element bodies. With element-based loads you must provide the element number (or an element set name) and the distributed load type label. The load type label identifies the type of load and the element face or edge on which the load is prescribed (see *About the Element Library* for definitions of the distributed load types available for particular elements). This method of specifying distributed loads is very general and can be used for all distributed load types and elements.

Surface-Based Loads

Use surface-based loads to prescribe a distributed load on a geometric surface or geometric edge. With surface-based loads you must specify the surface or edge name and the distributed load type. The surface or edge, which contains the element and face information, is defined as described in *Element-Based Surface Definition*. In Abaqus/CAE surfaces can be defined as collections of geometric faces and edges or collections of element faces and edges.This method of prescribing a distributed load facilitates user input for complex models. It can be used with most element types for which a valid surface can be defined. You can specify in the surface definition how the distributed load is applied to the boundary of an adaptive mesh domain in Abaqus/Explicit (see *Defining ALE Adaptive Mesh Domains in Abaqus/Explicit*).

Varying the Magnitude of a Load

The magnitude of a load is usually defined by the input data. The variation of the load magnitude during a step can be defined by the default amplitude variation for the step (see *About Prescribed Conditions*); by a user-defined amplitude curve (see *Amplitude Curves*); or, in some cases, by user subroutine *DLOAD*, *UDECURRENT*, *UDSECURRENT*, *UTRACLOAD*, or *VDLOAD*.

Loading during General Analysis Steps

If the analysis consists of one step only, the loads are defined in that step. If there are several analysis steps, the definition of loading in each analysis step depends on whether that step and the previous steps are general analysis steps or linear perturbation steps. Loading during linear perturbation steps is discussed below.

In general analysis steps, load magnitudes must always be given as total values, not as changes in magnitude. Multiple definitions of the same load condition in the same step are applied additively. Element-based and surface-based distributed loads are considered independently. For example, element-based and surface-based pressures applied to an element face in the same step are added. A single redefinition of that same load condition in a subsequent step, however, replaces all the like definitions (same load option, same load type) given in previous steps according to the rules described in *Removing Loads* below.

Any combination of loads can be applied together during a step. For a linear step it is possible to analyze several load cases based on the same stiffness.

Modifying Loads

At each new step the loading can be either modified or completely redefined. To redefine a load, the node, element, node set, element set, or surface name (and for gravity and surface-based general surface traction loading, the load direction) must be specified in exactly the same way and the load type must be identical. For example, if a node is part of a loaded node set in one step and is loaded as an individual node (by listing its node number) in another step, the loads will be added.

All loads defined in previous steps remain unchanged unless they are redefined, except for submodel distributed surface loads (see *Surface-Based Submodeling* for details). When a load is left unchanged, the following rules apply:

- If no amplitude is associated with the load, the load remains constant at the magnitude associated with the end of the previous step.
- If an associated amplitude is specified in terms of total time, the load continues to follow the amplitude definition.
- If an associated amplitude is specified in terms of step time, the load remains constant at the magnitude associated with the end of the previous step unless the amplitude is specified in user subroutine *UAMP* (Abaqus/Standard) or *VUAMP* (Abaqus/Explicit), in which case the load is removed immediately.

If you apply multiple loads of the same type at the same node (and the same degree of freedom if the degree of freedom is part of the loading definition), element, node set, element set, or surface, you cannot modify these loads in the following steps; you need to remove the loads and respecify them.

Input File Usage: Use either of the following options to modify an existing load or to specify an additional load (**LOADING OPTION* represents any load type):

**LOADING OPTION*
**LOADING OPTION*, OP=MOD

Abaqus/CAE Usage: Load module: **Create Load** or **Load Manager**: **Edit**

Removing Loads

If you choose to remove any load of a particular type (concentrated load, element-based distributed load, surface-based distributed load, etc.) in a step, no loads of that type will be propagated from the previous general step. All loads of that type that are in effect during this step must be respecified. To redefine a load, the node, element, node set, element set, or surface name must be specified in exactly the same way and the load type must be identical. Refer to *About Prescribed Conditions* for a discussion of amplitude variations when removing loads.

Input File Usage: Use the following option to release all previously applied loads of a given type and to specify new loads (**LOADING OPTION* represents any load type):

**LOADING OPTION*, OP=NEW

For example, **CLOAD*, OP=NEW with no data lines will remove all concentrated forces and moments from the model.

If the OP=NEW parameter is used on any loading option in a step, it must be used on all loading options of the same type within the step.

Abaqus/CAE Usage: Use the following option to remove a load within a step:

Load module: **Load Manager**: **Deactivate**

Abaqus/CAE automatically respecifies any loads that should remain in effect during this step.

Example

In the history definition input file section shown below, the distributed load (type BX) applied to element set A2 has a magnitude of 20.0 in the first step, which is changed to 50.0 in the second step. Both the set identifier (or element or node number) and the load type must be identical in both steps for Abaqus to identify a load for redefinition.

In Step 1 a concentrated load of magnitude 10.0 is applied to degree of freedom 3 of all nodes in node set NLEFT. In Step 2 a concentrated load of magnitude 5.0 is applied to degree of freedom 3 of node 1. If node 1 is in node set NLEFT, the total load applied in Step 2 at this node is 15.0: the loads add.

The two distributed loads of type P1 acting on element set E1 in Step 1 will be added to give a total distributed load of 43.0.

The pressure loads on element sets B3 and E1 are active during both steps.

```
*STEP
 Step 1
*STATIC
*CLOAD
 NLEFT, 3, 10.
*DLOAD
 A2, BX, 20.
 B3, P1, 5.
 E1, P1, 21.
*DLOAD
 E1, P1, 22.
*END STEP
**
*STEP
 Step 2
*STATIC
*CLOAD
 1, 3, 5.
*DLOAD, OP=MOD
 A2, BX, 50.
*END STEP
```

External work calculations when removing loads in a static analysis

External work calculations are approximate if a load is removed in a static step that is defined with a (nondefault) step amplitude. This is because a static analysis does not account for transient effects associated with the sudden removal of loads. Therefore, the total energy might be nonzero following load removal.

In this situation, you can obtain a better approximation to the external work by redefining such loads, but with a zero magnitude. In the latter case, you must ensure that you redefine the load in exactly the same way (that is, using the same node, element, node set, element set, or surface, as well as the load label) as it was defined in prior steps.

In the following example, a load of magnitude 100.0 is applied in the 3-direction to all the nodes in node set NSET_LOAD during Step 1. The load is removed in Step 2. The redefinition of the load with a zero magnitude is optional; however, it ensures accurate external work calculations. If accuracy of the external work calculations is not a concern, you do not need to redefine the load.

```
*STEP
*STATIC
*CLOAD
NSET_LOAD, 3, 100.
…
```

```
*END STEP
*STEP, AMP=STEP
*STATIC
*CLOAD, OP=NEW
 ** The redefinition of the load with a zero magnitude is optional, and
added here
**  to ensure more accurate external work calculations.
NSET_LOAD, 3, 0.
…
*END STEP
```

Follower Loads in Large-Displacement Analysis

In large-displacement analysis distributed loads will be treated as follower forces when appropriate. For beam and shell elements point (concentrated) loads may be fixed in direction or they may rotate with the structure depending on whether you specify follower forces for the load (see *Concentrated Loads*). Follower loads defined at a rigid body tie node rotate with the rigid body in Abaqus/Explicit.

Loading during Linear Perturbation Steps

In a linear perturbation step (available only in Abaqus/Standard) the state at the end of the previous general analysis step is considered as the "base state." If the linear perturbation step is the first step of the analysis, the initial conditions of the model form the base state. Loading during a linear perturbation step must be defined as the change in load from the base state (the perturbation of load), not the total of the base state load plus the perturbation load.

In consecutive linear perturbation steps, the perturbation of load that applies to each step must be defined completely within that step—the analysis within each such step always starts from the base state (except when you specify that a modal dynamic step should use the initial conditions from the immediately preceding step—see *Transient Modal Dynamic Analysis*).

In nonlinear steps that follow linear perturbation analysis steps, the analysis is continued from the base state as if the intermediate linear perturbation steps did not exist.

Loading during Linear (Mode-Based) Dynamics Procedures

If a user subroutine is used to define loading in a mode-based linear dynamics analysis, the subroutine will be called only at the beginning of the step to obtain the magnitude of the load. The load magnitude then remains constant in the step unless it is modified by an amplitude curve.

Concentrated Loads

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Loads*
- **CLOAD*
- *Defining a concentrated force*
- *Defining a moment*
- *Defining a generalized plane strain load*

Overview

You can apply concentrated forces or moments at any nodal degree of freedom.

Concentrated loads:

- can be fixed in direction; or
- can rotate as the node rotates (referred to as follower forces), resulting in an additional, and possibly unsymmetric, contribution to the load stiffness.

In steady-state dynamic analysis both real and imaginary concentrated loads can be applied (see *Direct-Solution Steady-State Dynamic Analysis* and *Mode-Based Steady-State Dynamic Analysis* for details).

Multiple concentrated load cases can be defined in random response analysis (see *Random Response Analysis* for details).

Concentrated loads are also used to apply the pressure-conjugate at nodes with pressure degree of freedom in acoustic analysis (see *Acoustic and Shock Loads*).

Actuation loads in connector elements can be defined as connector loads, applied similarly to concentrated loads. See *Connector Actuation* for more detailed information.

The procedures in which these loads can be used are outlined in *About Prescribed Conditions*. See *About Loads* for general information that applies to all types of loading.

Concentrated Loads

In Abaqus/Standard and Abaqus/Explicit analyses concentrated forces or moments can be applied at any nodal degree of freedom.

You should not apply a moment load at the origin of a cylindrical coordinate system; doing so would make the radial and tangential loads indeterminate.

Input File Usage: **CLOAD*
node number or node set, degree of freedom, magnitude

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Concentrated force**, **Moment**, or **Generalized plane strain** for the **Types for Selected Step**

Specifying Concentrated Follower Forces

You can specify that the direction of a concentrated force should rotate with the node to which it is applied. This specification should be used only in large-displacement analysis and can be used only at nodes with active rotational degrees of freedom (such as the nodes of beam and shell elements or, in Abaqus/Explicit, tie nodes on a rigid body), excluding the reference node of generalized plane strain elements. If you specify follower forces, the components of the concentrated force must be specified with respect to the reference configuration.

Follower loads lead to an unsymmetric contribution to the stiffness matrix that is generally referred to as the load stiffness. Some issues associated with the load stiffness contribution are discussed in *Improving the Rate of Convergence in Large-Displacement Implicit Analysis*.

Input File Usage: **CLOAD*, FOLLOWER

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Concentrated force** or **Moment** for the **Types for Selected Step**: **Follow nodal rotation**

Defining the Values of Concentrated Nodal Force from a User-Specified File

You can define nodal force using nodal force output from a particular step and increment in the output database (`.odb`) file of a previous Abaqus analysis. The part (`.prt`) file from the original analysis is also required when reading data from the output database file. In this case both the previous model and the current model must be defined consistently, including node numbering, which must be the same in both models. If the models are defined in terms of an assembly of part instances, part instance naming must be the same.

Input File Usage: **CLOAD*, FILE=file, STEP=step, INC=inc

Abaqus/CAE Usage: Defining the values of concentrated nodal force from a user-specified file is not supported in Abaqus/CAE.

Defining Time-Dependent Concentrated Loads

The prescribed magnitude of a concentrated load can vary with time during a step according to an amplitude definition, as described in *About Prescribed Conditions*. If different variations are needed for different loads, each load can refer to its own amplitude.

Modifying Concentrated Loads

Concentrated loads can be added, modified, or removed as described in *About Loads*.

Improving the Rate of Convergence in Large-Displacement Implicit Analysis

When concentrated follower forces are specified in a geometrically nonlinear static and dynamic analysis, the unsymmetric matrix storage and solution scheme should normally be used. See *Defining an Analysis* for more information on the unsymmetric matrix storage and solution scheme.

Distributed Loads

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Loads*
- *Fluid Pressure Penetration Loads*
- *Dynamic Stress/Displacement Analysis*
- **DLOAD*
- **DSLOAD*
- *Defining a pressure load*
- *Defining a shell edge load*
- *Defining a pipe pressure load*
- *Defining a body force*
- *Defining a line load*
- *Defining a gravity load*
- *Defining a rotational body force*

Overview

You can prescribe distributed loads on element faces, element bodies, or element edges and over geometric surfaces or geometric edges.

Distributed loads:

- require that an appropriate distributed load type be specified—see *About the Element Library* for definitions of the distributed load types available for particular elements; and
- can be of follower type, which can rotate during a geometrically nonlinear analysis and result in an additional (often unsymmetric) contribution to the stiffness matrix that is generally referred to as the load stiffness.

The procedures in which these loads can be used are outlined in *About Prescribed Conditions*. See *About Loads* for general information that applies to all types of loading.

Follower loads are discussed further in *Follower Surface Loads* and *Follower Edge and Line Loads*. The contribution of follower loads to load stiffness is discussed in *Improving the Rate of Convergence in Large-Displacement Implicit Analysis*.

In steady-state dynamic analysis both real and imaginary distributed loads can be applied (see *Direct-Solution Steady-State Dynamic Analysis* and *Mode-Based Steady-State Dynamic Analysis* for details).

Incident wave loading is used to apply distributed loads for the special case of loads associated with a wave traveling through an acoustic medium. Inertia relief is used to apply inertia-based loading in Abaqus/Standard. These load types are discussed in *Acoustic and Shock Loads* and *Inertia Relief*, respectively. Abaqus/Aqua load types are discussed in *Abaqus/Aqua Analysis*.

Defining Time-Dependent Distributed Loads

The prescribed magnitude of a distributed load can vary with time during a step according to an amplitude definition, as described in *About Prescribed Conditions*. If different variations are required for different loads, each load can refer to its own amplitude definition.

Modifying Distributed Loads

Distributed loads can be added, modified, or removed as described in *About Loads*.

Improving the Rate of Convergence in Large-Displacement Implicit Analysis

In large-displacement analyses in Abaqus/Standard some distributed load types introduce unsymmetric load stiffness matrix terms. Examples are hydrostatic pressure, pressure applied to surfaces with free edges, Coriolis force, rotary acceleration force, and distributed edge loads and surface tractions modeled as follower loads. In such cases using the unsymmetric matrix storage and solution scheme for the analysis step might improve the convergence rate of the equilibrium iterations. See *Defining an Analysis* for more information on the unsymmetric matrix storage and solution scheme.

Defining Distributed Loads in a User Subroutine

Nonuniform distributed loads such as a nonuniform body force in the *X*-direction can be defined by means of user subroutine *DLOAD* in Abaqus/Standard or *VDLOAD* in Abaqus/Explicit. When an amplitude reference is used with a nonuniform load defined in user subroutine *VDLOAD*, the current value of the amplitude function is passed to the user subroutine at each time increment in the analysis. *DLOAD* and *VDLOAD* are not available for surface tractions, edge tractions, or edge moments.

In Abaqus/Standard nonuniform distributed surface tractions, edge tractions, and edge moments can be defined by means of user subroutine *UTRACLOAD*. User subroutine *UTRACLOAD* allows you to define a nonuniform magnitude for surface tractions, edge tractions, and edge moments, as well as nonuniform loading directions for general surface tractions, shear tractions, and general edge tractions.

Nonuniform distributed surface tractions, edge tractions, and edge moments are not currently supported in Abaqus/Explicit.

When the user subroutine is used, the external work is calculated based only on the current magnitude of the distributed load since the incremental value for the distributed load is not defined.

Specifying the Region to Which a Distributed Load Is Applied

As discussed in *About Loads*, distributed loads can be defined as element-based or surface-based. Element-based distributed loads can be prescribed on element bodies, element surfaces, or element edges. Surface-based distributed loads can be prescribed directly on geometric surfaces or geometric edges.

Three types of distributed loads can be defined: body loads, surface loads, and edge loads. Distributed body loads are always element-based. Distributed surface loads and distributed edge loads can be element-based or surface-based. The regions on which each load type can be prescribed are summarized in *Table 1* and *Table 2*. In Abaqus/CAE distributed loads are specified by selecting the region in the viewport or from a list of surfaces. In the Abaqus input file different options are used depending on the type of region to which the load is applied, as illustrated in the following sections.

Table 1: Regions on which the different load types can be prescribed.

Load type	Load definition	Input file region
Body loads	Element-based	Element bodies
Surface loads	Element-based	Element surfaces
	Surface-based	Geometric element-based surfaces
Edge loads (including beam line loads)	Element-based	Element edges
	Surface-based	Geometric edge-based surfaces

Table 2: Regions in Abaqus/CAE on which the different load types can be prescribed.

Load type	Load definition	Abaqus/CAE region
Body loads	Element-based	Volumetric bodies
Surface loads	Element-based	Surfaces defined as collections of geometric faces or element faces (excluding analytical rigid surfaces)
	Surface-based	
Edge loads (including beam line loads)	Element-based	Surfaces defined as collections of geometric edges or element edges
	Surface-based	

Applying Electric Machine Loads

You can apply loads from one or more two-dimensional electromagnetic analyses to the stator surface of an electric machine in Abaqus/Standard to perform a three-dimensional noise and vibration analysis.

You must specify the stator three-dimensional surface name, the number of electric machine teeth, and the number of surface slices on which the electromagnetic analyses were conducted. In addition, you must specify the stator orientation and the width of each slice.

Abaqus/Standard expects the electromagnetic loads in the form of radial and tangential two-dimensional forces and compensating torque at a certain operating point of the electrical machine. The electric machine operating point is defined by the rotor torque and rotational speed (revolutions per time). You must specify each of these forces and torques at certain space and time orders for each slice.

You can specify the electric machine load as a separate single load or inside a load case.

Abaqus/Standard converts the electromagnetic forces and torques internally into distributed surface loads and concentrated force loads on the stator cylindrical surfaces of the electric machine. It also splits the user-specified stator surface internally into subsurfaces on which these internally converted loads are applied. Abaqus/Standard bases the subsurfaces creation on the number of teeth and the number of slices. The loads on the subsurfaces of a given slice differ only in phase.

Input File Usage: Use the following option in the model data to define an electric machine property:

**ELECTRIC MACHINE PROPERTY*, NAME=*EMP-Name*,
SURFACE=*Surface-Name*

Input File Usage: Use the following option in the step data to define an electric machine load:

**ELECTRIC MACHINE LOAD*, PROPERTY=*EMP-Name*

Body Forces

Body loads, such as gravity, centrifugal, Coriolis, and rotary acceleration loads, are applied as element-based loads. The units of a body force are force per unit volume.

The distributed body load types that are available in Abaqus, along with the corresponding load type labels, are listed in *Table 3* and *Table 4*.

Table 3: Distributed body load types.

Load description	Load type label for element-based loads
Body force in global *X*-, *Y*-, and *Z*-directions	BX, BY, BZ
Nonuniform body force in global *X*-, *Y*-, and *Z*-directions	BXNU, BYNU, BZNU
Body force in radial and axial directions (only for axisymmetric elements)	BR, BZ
Nonuniform body force in radial and axial directions (only for axisymmetric elements)	BRNU, BZNU
Viscous body force in global *X*-, *Y*-, and *Z*-directions (available only in Abaqus/Explicit)	VBF
Stagnation body force in global *X*-, *Y*-, and *Z*-directions (available only in Abaqus/Explicit)	SBF
Gravity loading	GRAV
Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume and ω is the angular velocity)	CENT
Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity)	CENTRIF
Coriolis force	CORIO
Rotary acceleration load	ROTA
Rotordynamic load	ROTDYNF
Porous drag load (input is porosity of the medium)	PDBF

Table 4: Distributed body load types in Abaqus/CAE.

Load description	Abaqus/CAE load type
Body force in global *X*-, *Y*-, and *Z*-directions	**Body force**
Nonuniform body force in global *X*-, *Y*-, and *Z*-directions	**Body force**
Body force in radial and axial directions (only for axisymmetric elements)	
Nonuniform body force in radial and axial directions (only for axisymmetric elements)	
Viscous body force in global *X*-, *Y*-, and *Z*-directions (available only in Abaqus/Explicit)	Not supported
Stagnation body force in global *X*-, *Y*-, and *Z*-directions (available only in Abaqus/Explicit)	
Gravity loading	**Gravity**
Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume and ω is the angular velocity)	Not supported
Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity)	**Rotational body force**

Load description	Abaqus/CAE load type
Coriolis force	**Coriolis force**
Rotary acceleration load	**Rotational body force**
Rotordynamic load	**Rotordynamic load**
Porous drag load (input is porosity of the medium)	**Porous drag body force**

Specifying General Body Forces

You can specify body forces on any elements in the global *X*-, *Y*-, or *Z*-direction. You can specify body forces on axisymmetric elements in the radial or axial direction.

Input File Usage: Use the following option to define a body force in the global *X*-, *Y*-, or *Z*-direction:

**DLOAD*
element number or element set, *load type label*, *magnitude*

where *load type label* is BX, BY, BZ, BXNU, BYNU, or BZNU.

Use the following option to define a body force in the radial or axial direction on axisymmetric elements:

**DLOAD*
element number or element set, *load type label*, *magnitude*

where *load type label* is BR, BZ, BRNU, or BZNU.

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Body force** for the **Types for Selected Step**

Specifying Viscous Body Force Loads in Abaqus/Explicit

Viscous body force loads are defined by

$$\mathbf{f_v} = -\mathbf{c_{vb}}\,(\mathbf{v} - \mathbf{v_{ref}})\,\mathbf{V_e},$$

where $\mathbf{f_v}$ is the viscous force applied to the body; c_{vb} is the coefficient of viscosity, given as the magnitude of the load; $\mathbf{v}$ is the velocity of the point on the body where the force is being applied; $\mathbf{v}_{ref}$ is the velocity of the reference node; and V_e is the element volume.

Viscous body force loading can be thought of as mass-proportional damping in the sense that it gives a damping contribution proportional to the mass for an element if the coefficient c_{vb} is chosen to be a small value multiplied by the material density ρ (see *Material Damping*). Viscous body force loading provides an alternative way to define mass-proportional damping as a function of relative velocities and a step-dependent damping coefficient.

Input File Usage: Use the following option to define a viscous body force load:

**DLOAD*, REF NODE=*reference_node*
element number or element set, VBF, *magnitude*

Abaqus/CAE Usage: Viscous body force loads are not supported in Abaqus/CAE.

Specifying Stagnation Body Force Loads in Abaqus/Explicit

Stagnation body force loads are defined by

$$\mathbf{f_s} = -\mathbf{c_{sb}}(\mathbf{v} - \mathbf{v_{ref}})^2 \mathbf{V_e},$$

where $\mathbf{f_s}$ is the stagnation body force applied to the body; c_{sb} is the factor, given as the magnitude of the load; $\mathbf{v}$ is the velocity of the point on the body where the body force is being applied; $\mathbf{v}_{ref}$ is the velocity of the reference node; and V_e is the element volume. The coefficient c_{sb} should be very small to avoid excessive damping and a dramatic drop in the stable time increment.

Input File Usage: Use the following option to define a stagnation body force load:

**DLOAD*, REF NODE=*reference_node*
element number or element set, SBF, *magnitude*

Abaqus/CAE Usage: Stagnation body force loads are not supported in Abaqus/CAE.

Specifying Gravity Loading

Gravity loading (uniform acceleration in a fixed direction) is specified by using the gravity distributed load type and giving the actual magnitude of the load. The direction of the gravity field is specified by giving the components of the gravity vector in the distributed load definition. Abaqus uses the user-specified material density (see *Density*), together with the magnitude and direction, to calculate the loading. The magnitude of the gravity load can vary with time during a step according to an amplitude definition, as described in *About Prescribed Conditions*. However, the direction of the gravity field is always applied at the beginning of the step and remains fixed during the step.

The gravity load can be applied automatically to the entire model. Omit the element number or element set to automatically collect all elements in the model that have mass contributions (including point mass elements but excluding rigid elements) in an element set called `_Whole_Model_GRAV_Elset`, and apply the gravity loads to the elements in this element set.

When gravity loading is used with substructures, the density must be defined and unit gravity load vectors must be calculated when the substructure is created (see *Generating Substructures*).

For beam elements the resultant force for gravity loading is always applied such that it passes through the origin of the beam section's local coordinate system, independent of the location of this origin relative to the centroid of the section.

Input File Usage: Use the following option to define a gravity load:

**DLOAD*
element number or element set (or blank), GRAV, *actual magnitude of gravity load*, *comp1, comp2, comp3*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Gravity** for the **Types for Selected Step**

Specifying Loads due to Rotation of the Model in Abaqus/Standard

Centrifugal loads, Coriolis forces, rotary acceleration, and rotordynamic loads can be applied in Abaqus/Standard by specifying the appropriate distributed load type in an element-based distributed load definition. These loading options are primarily intended for replicating dynamic loads while performing analyses other than implicit dynamics using direct integration (*Dynamic Stress/Displacement Analysis*). In an implicit dynamic procedure inertia loads due to rotations come about naturally due to the equations of motion. Applying distributed centrifugal, Coriolis, rotary acceleration, and rotordynamic loads in an implicit dynamic analysis might lead to nonphysical loads and should be used carefully.

These loads can be applied automatically to the entire model. Omit the element number or element set to automatically collect all applicable elements in the model into an element set called `_Whole_Model_xxx_Elset`, where *xxx* is the load type, and apply the load to the elements in this element set.

Centrifugal Loads

Centrifugal load magnitudes can be specified as ω^2, where ω is the angular velocity in radians per time. Abaqus/Standard uses the specified material density (see *Density*), together with the load magnitude and the axis of rotation, to calculate the loading. Alternatively, a centrifugal load magnitude can be given as $\rho\omega^2$, where ρ is the material density (mass per unit volume) for solid or shell elements or the mass per unit length for beam elements and ω is the angular velocity in radians per time. This type of centrifugal load formulation does not account for large volume changes. The two centrifugal load types will produce slightly different local results for first-order elements; $\rho\omega^2$ uses a consistent mass matrix, and ω^2 uses a lumped mass matrix in calculating the load forces and load stiffnesses. The output variables for these two centrifugal load types are CENTMAG and CENTRIFMAG, respectively.

The magnitude of the centrifugal load can vary with time during a step according to an amplitude definition, as described in *About Prescribed Conditions*. However, the position and orientation of the axis around which the structure rotates, which is defined by giving a point on the axis and the axis direction, are always applied at the beginning of the step and remain fixed during the step.

Input File Usage: Use either of the following options to define a centrifugal load:

**DLOAD*

element number or element set (or blank), CENTRIF, ω^2, *coord1, coord2, coord3, comp1,*
comp2, comp3
**DLOAD*

element number or element set (or blank), CENT, $\rho\omega^2$, *coord1, coord2, coord3, comp1,*
comp2, comp3

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Rotational body force** for the **Types for Selected Step**: **Load effect: Centrifugal**

Coriolis Forces

Coriolis force is defined by specifying the Coriolis distributed load type and giving the load magnitude as $\rho\omega$, where ρ is the material density (mass per unit volume) for solid and shell elements or the mass per unit length

for beam elements and ω is the angular velocity in radians per time. The magnitude of the Coriolis load can vary with time during a step according to an amplitude definition, as described in *About Prescribed Conditions*. However, the position and orientation of the axis around which the structure rotates, which is defined by giving a point on the axis and the axis direction, are always applied at the beginning of the step and remain fixed during the step.

In a static analysis Abaqus computes the translational velocity term in the Coriolis loading by dividing the incremental displacement by the current time increment.

The Coriolis load formulation does not account for large volume changes.

Input File Usage: Use the following option to define a Coriolis load:

**DLOAD*
element number or element set (or blank), CORIO, $\rho\omega$, *coord1, coord2, coord3, comp1, comp2, comp3*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Coriolis force** for the **Types for Selected Step**

Rotary Acceleration Loads

Rotary acceleration loads are defined by specifying the rotary acceleration distributed load type and giving the rotary acceleration magnitude, α, in radians/time^2, which includes any precessional motion effects. The axis of rotary acceleration must be defined by giving a point on the axis and the axis direction. Abaqus/Standard uses the specified material density (see *Density*), together with the rotary acceleration magnitude and axis of rotary acceleration, to calculate the loading. The magnitude of the load can vary with time during a step according to an amplitude definition, as described in *About Prescribed Conditions*. However, the position and orientation of the axis around which the structure rotates are always applied at the beginning of the step and remain fixed during the step.

Rotary acceleration loads are not applicable to axisymmetric elements.

Input File Usage: Use the following option to define a rotary acceleration load:

**DLOAD*
element number or element set (or blank), ROTA, α, *coord1, coord2, coord3, comp1, comp2, comp3*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Rotational body force** for the **Types for Selected Step**: **Load effect: Rotary acceleration**

Specifying General Rigid-Body Acceleration Loading in Abaqus/Standard

General rigid-body acceleration loading can be specified in Abaqus/Standard by using a combination of the gravity, centrifugal (ω^2), and rotary acceleration load types.

Rotordynamic Loads in a Fixed Reference Frame

Rotordynamic loads can be used to study the vibrational response of three-dimensional models of axisymmetric structures, such as a flywheel in a hybrid energy storage system, that are spinning about their axes of symmetry in a fixed reference frame (see *Genta, 2005*). This is in contrast to the centrifugal loads, Coriolis forces, and

rotary acceleration loads discussed above, which are formulated in a rotating frame. Rotordynamic loads are, therefore, not intended to be used in conjunction with these other dynamic load types.

The intended workflow for rotordynamic loads is to define the load in a nonlinear static step to establish the centrifugal load effects and load stiffness terms associated with a spinning body. The nonlinear static step can then be followed by a sequence of linear dynamic analyses such as complex eigenvalue extraction and/or a subspace or direct-solution steady-state dynamic analysis to study complex dynamic behaviors (induced by gyroscopic moments) such as critical speeds, unbalanced responses, and whirling phenomena in rotating structures. You do not need to redefine the rotordynamic load in the linear dynamic analyses—the load definition is carried over from the nonlinear static step. The contribution of the gyroscopic matrices in the linear dynamic steps is unsymmetric; therefore, you must use unsymmetric matrix storage as described in *Defining an Analysis* during these steps.

Rotordynamic loads are intended only for three-dimensional models of axisymmetric bodies; you must ensure that this modeling assumption is met. Rotordynamic loads are supported for all three-dimensional continuum and cylindrical elements, shell elements, membrane elements, cylindrical membrane elements, beam elements, and rotary inertia elements. The spinning axis defined as part of the load must be the axis of symmetry for the structure. Therefore, beam elements must be aligned with the symmetry axis. In addition, one of the principal directions of each loaded rotary inertia element must be aligned with the symmetry axis, and the inertia components of the rotary inertia elements must be symmetric about this axis. Multiple spinning structures spinning about different axes can be modeled in the same step. The spinning structures can also be connected to nonaxisymmetric, nonrotating structures (such as bearings or support structures).

Rotordynamic loads are defined by specifying the angular velocity, $\boldsymbol{\omega}$, in radians per time. The magnitude of the rotordynamic load can vary with time during a step according to an amplitude definition, as described in *About Prescribed Conditions*. However, the position and orientation of the axis around which the structure rotates, which is defined by giving a point on the axis and the axis direction, are always applied at the beginning of the step and remain fixed during the step.

Input File Usage: Use the following option to define a rotordynamic load:

**DLOAD*
element number or element set (or blank), ROTDYNF, $\boldsymbol{\omega}$, *coord1, coord2, coord3,*

comp1, comp2, comp3

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Rotational body force** for the **Types for Selected Step**: **Load effect: Rotordynamic load**

Surface Tractions and Pressure Loads

General or shear surface tractions and pressure loads can be applied in Abaqus as element-based or surface-based distributed loads. The units of these loads are force per unit area.

The distributed surface load types that are available in Abaqus, along with the corresponding load type labels, are listed in *Table 5* and *Table 6*. *About the Element Library* lists the distributed surface load types that are available for particular elements and the Abaqus/CAE load support for each load type. For some element-based loads you must identify the face of the element upon which the load is prescribed in the load type label (for example, P*n* or P*n*NU for continuum elements).

Table 5: Distributed surface load types.

Load description	Load type label for element-based loads	Load type label for surface-based loads
General surface traction	TRVEC*n*, TRVEC	TRVEC
Shear surface traction	TRSHR*n*, TRSHR	TRSHR
Nonuniform general surface traction	TRVEC*n*NU, TRVECNU	TRVECNU
Nonuniform shear surface traction	TRSHR*n*NU, TRSHRNU	TRSHRNU
Pressure	P*n*, P	P
Nonuniform pressure	P*n*NU, PNU	PNU
Fluid pressure penetration	Not applicable	PPEN
Hydrostatic pressure (available only in Abaqus/Standard)	HP*n*, HP	HP
Viscous pressure (available only in Abaqus/Explicit)	VP*n*, VP	VP
Stagnation pressure (available only in Abaqus/Explicit)	SP*n*, SP	SP
Surface tension (available only in Abaqus/Explicit)	Not applicable	ST
Pore mechanical pressure (available only in Abaqus/Standard)	PORMECH*n*, PORMECH	PORMECH
Hydrostatic internal and external pressure (only for PIPE and ELBOW elements)	HPI, HPE	Not applicable
Uniform internal and external pressure (only for PIPE and ELBOW elements)	PI, PE	Not applicable
Nonuniform internal and external pressure (only for PIPE and ELBOW elements)	PINU, PENU	Not applicable
Nodal pressure (available only in Abaqus/Standard)	Not applicable	NP

Table 6: Distributed surface load types in Abaqus/CAE.

Load description	Abaqus/CAE load type
General surface traction	**Surface traction**
Shear surface traction	
Nonuniform general surface traction	**Surface traction** (surface-based loads only)
Nonuniform shear surface traction	
Pressure	**Pressure**
Nonuniform pressure	**Pressure** (surface-based loads only)
Hydrostatic pressure (available only in Abaqus/Standard)	
Viscous pressure (available only in Abaqus/Explicit)	
Stagnation pressure (available only in Abaqus/Explicit)	
Hydrostatic internal and external pressure (only for PIPE and ELBOW elements)	**Pipe pressure**
Uniform internal and external pressure (only for PIPE and ELBOW elements)	
Nonuniform internal and external pressure (only for PIPE and ELBOW elements)	

Follower Surface Loads

By definition, the line of action of a *follower* surface load rotates with the surface in a geometrically nonlinear analysis. This is in contrast to a *nonfollower* load, which always acts in a fixed global direction.

With the exception of general surface tractions, all the distributed surface loads listed in *Table 5* and *Table 6* are modeled as follower loads. The hydrostatic and viscous pressures listed in *Table 5* and *Table 6* always act normal to the surface in the current configuration, the shear tractions always act tangent to the surface in the current configuration, and the internal and external pipe pressures follow the motion of the pipe elements.

General surface tractions can be specified to be follower or nonfollower loads. There is no difference between a follower and a nonfollower load in a geometrically linear analysis since the configuration of the body remains fixed. The difference between a follower and nonfollower general surface traction is illustrated in the next section through an example.

Input File Usage: Use one of the following options to define general surface tractions as follower loads (the default):

**DLOAD*, FOLLOWER=YES
**DSLOAD*, FOLLOWER=YES

Use one of the following options to define general surface tractions as nonfollower loads:

**DLOAD*, FOLLOWER=NO
**DSLOAD*, FOLLOWER=NO

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction: General**, toggle on or off **Follow rotation**

Specifying General Surface Tractions

General surface tractions allow you to specify a surface traction, $\mathbf{t}$, acting on a surface S. The resultant load, $\mathbf{f}$, is computed by integrating $\boldsymbol{t}$ over S:

$$\mathbf{f} = \int_S \mathbf{t}\, dS = \int_S \alpha\, \hat{\mathbf{t}}\, dS,$$

where α is the magnitude and $\hat{\mathbf{t}}$ is the direction of the load. To define a general surface traction, you must specify both a load magnitude, α, and the direction of the load with respect to the reference configuration, $\mathbf{t}_{user}$. The magnitude and direction can also be specified in user subroutine *UTRACLOAD*. The specified traction directions are normalized by Abaqus and, thus, do not contribute to the magnitude of the load:

$$\hat{\mathbf{t}}^o = \frac{\mathbf{t}_{user}}{\|\mathbf{t}_{user}\|}.$$

Input File Usage: Use one of the following options to define a general surface traction:

**DLOAD*
element number or element set, *load type label*, *magnitude*,
direction components

where *load type label* is TRVEC*n*, TRVEC, TRVEC*n*NU, or TRVECNU.

**DSLOAD*
surface name, TRVEC or TRVECNU, *magnitude*, *direction components*

Abaqus/CAE Usage: Use the following input to define an element-based general surface traction:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction: General**, **Distribution**: select an analytical field

Use the following input to define a surface-based general surface traction:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction: General**, **Distribution**: **Uniform** or **User-defined**

Nonuniform element-based general surface traction is not supported in Abaqus/CAE.

Defining the Direction Vector with Respect to a Local Coordinate System

By default, the components of the traction vector are specified with respect to the global directions. You can also refer to a local coordinate system (see *Orientations*) for the direction components of these tractions. See *Examples: Using a Local Coordinate System to Define Shear Directions* below for an example of a traction load defined with respect to a local coordinate system. When using local coordinate systems for tractions applied to two-dimensional solid elements, you must ensure that the nonzero components of the loads are applied only in the *X*- and *Y*-directions. Traction loads in the third direction are not supported (*Z*-direction for plane strain and plane stress elements, θ-direction for axisymmetric elements).

Input File Usage: Use one of the following options to specify a local coordinate system:

**DLOAD*, ORIENTATION=*name*
**DSLOAD*, ORIENTATION=*name*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: select **CSYS: Picked** and click **Edit** to pick a local coordinate system, or select **CSYS: User-defined** to enter the name of a user subroutine that defines a local coordinate system

Rotation of the Traction Vector Direction

The traction load acts in the fixed direction $\hat{\mathbf{t}} = \hat{\mathbf{t}}^o$ in a geometrically linear analysis or if a nonfollower load is specified in a geometrically nonlinear analysis (which includes a perturbation step about a geometrically nonlinear base state).

If a follower load is specified in a geometrically nonlinear analysis, the traction load rotates rigidly with the surface using the following algorithm. The reference configuration traction vector, $\mathbf{t}^o = \alpha \hat{\mathbf{t}}^o$, is decomposed by Abaqus into two components: a normal component,

$$\alpha \hat{\mathbf{t}}^o \cdot \mathbf{N}\mathbf{N} = \alpha_n \mathbf{N},$$

and a tangential component,

$$\alpha \left(\hat{\mathbf{t}}^o - \hat{\mathbf{t}}^o \cdot \mathbf{N}\mathbf{N} \right) = \alpha_d \mathbf{D},$$

where $\mathbf{N}$ is the unit reference surface normal and $\mathbf{D}$ is the unit projection of $\hat{\mathbf{t}}^o$ onto the reference surface. The applied traction in the current configuration is then computed as

$$\mathbf{t} = \alpha_n \mathbf{n} + \alpha_d \mathbf{d},$$

where $\mathbf{n}$ is the normal to the surface in the current configuration and $\mathbf{d}$ is the image of $\mathbf{D}$ rotated onto the current surface; that is, $\mathbf{d} = \mathbf{RD}$, where $\mathbf{R}$ is the standard rotation tensor obtained from the polar decomposition of the local two-dimensional surface deformation gradient $\mathbf{F} = \mathbf{RU}$.

Examples: Follower and Nonfollower Tractions

The following two examples illustrate the difference between applying follower and nonfollower tractions in a geometrically nonlinear analysis. Both examples refer to a single 4-node plane strain element (element 1). In Step 1 of the first example a follower traction load is applied to face 1 of element 1, and a nonfollower traction load is applied to face 2 of element 1. The element is rotated rigidly 90° counterclockwise in Step 1 and then another 90° in Step 2. As illustrated in *Figure 1*, the follower traction rotates with face 1, while the nonfollower traction on face 2 always acts in the global *x*-direction.

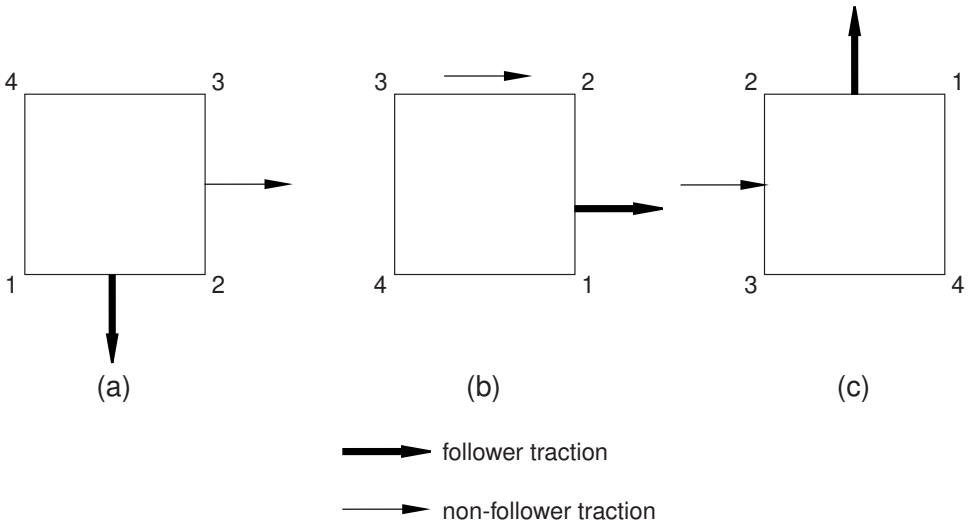


Figure 1: Follower and nonfollower traction loads in a geometrically nonlinear analysis, load applied in Step 1: (a) beginning of Step 1; (b) end of Step 1, beginning of Step 2; (c) end of Step 2.

```
*STEP, NLGEOM
 Step 1 - Rotate square 90 degrees
...
*DLOAD, FOLLOWER=YES
 1, TRVEC1, 1., 0., -1., 0.
*DLOAD, FOLLOWER=NO
 1, TRVEC2, 1., 1., 0., 0.
*END STEP
*STEP, NLGEOM
 Step 2 - Rotate square another 90 degrees
...
*END STEP
```

In the second example the element is rotated 90° counterclockwise with no load applied in Step 1. In Step 2 a follower traction load is applied to face 1, and a nonfollower traction load is applied to face 2. The element is then rotated rigidly by another 90°. The direction of the follower load is specified with respect to the original

configuration. As illustrated in *Figure 2*, the follower traction rotates with face 1, while the nonfollower traction on face 2 always acts in the global *x*-direction.

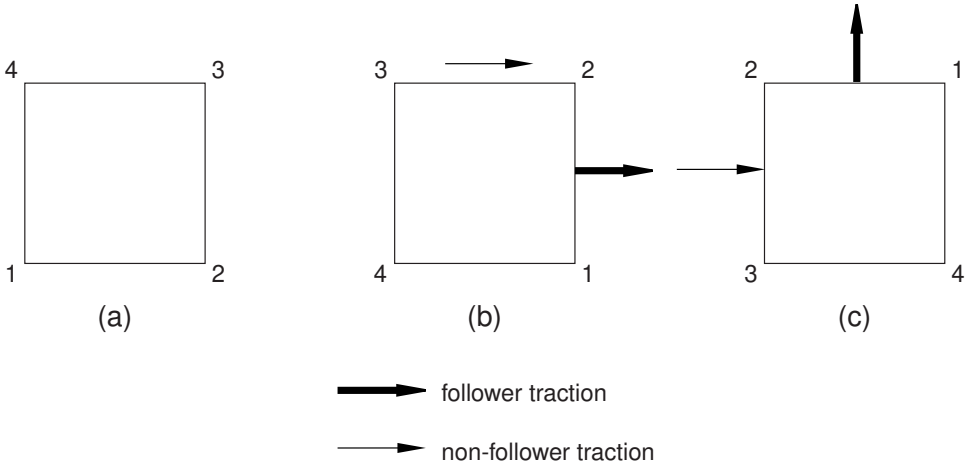


Figure 2: Follower and nonfollower traction loads in a geometrically nonlinear analysis, load applied in Step 2: (a) beginning of Step 1; (b) end of Step 1, beginning of Step 2; (c) end of Step 2.

```
*STEP, NLGEOM
 Step 1 - Rotate square 90 degrees
...
*END STEP
*STEP, NLGEOM
 Step 2 - Rotate square another 90 degrees
*DLOAD, FOLLOWER=YES
 1, TRVEC1, 1., 0., -1., 0.
*DLOAD, FOLLOWER=NO
 1, TRVEC2, 1., 1., 0., 0.
...
*END STEP
```

Specifying Shear Surface Tractions

Shear surface tractions allow you to specify a surface force per unit area, $\mathbf{t}_s$, that acts tangent to a surface S. The resultant load, $\mathbf{f}$, is computed by integrating $\mathbf{t}_s$ over S:

$$\mathbf{f} = \int_S \mathbf{t}_s \, dS = \int_S \alpha \, \mathbf{d} \, dS,$$

where α is the magnitude and $\mathbf{d}$ is a unit vector along the direction of the load. To define a shear surface traction, you must provide both the magnitude, α, and a direction, $\mathbf{t}_{user}$, for the load. The magnitude and direction vector can also be specified in user subroutine *UTRACLOAD*.

Abaqus modifies the traction direction by first projecting the user-specified vector, $\mathbf{t}_{user}$, onto the surface in the *reference* configuration,

$$\mathbf{t}_{user}^{po} = \mathbf{t}_{user} - \mathbf{t}_{user} \cdot \mathbf{NN},$$

where $\mathbf{N}$ is the reference surface normal. The specified traction is applied along the computed traction direction $\mathbf{D}$ tangential to the surface:

$$\mathbf{D} = \frac{\mathbf{t}_{user}^{po}}{\|\mathbf{t}_{user}^{po}\|}.$$

Consequently, a shear traction load is not applied at any point where $\mathbf{t}_{user}$ is normal to the reference surface.

The shear traction load acts in the fixed direction $\mathbf{d} = \mathbf{D}$ in a geometrically linear analysis. In a geometrically nonlinear analysis (which includes a perturbation step about a geometrically nonlinear base state), the shear traction vector will rotate rigidly; that is, $\mathbf{d} = \mathbf{RD}$, where $\mathbf{R}$ is the standard rotation tensor obtained from the polar decomposition of the local two-dimensional surface deformation gradient $\mathbf{F} = \mathbf{RU}$.

Input File Usage: Use one of the following options to define a shear surface traction:

**DLOAD*
element number or element set, *load type label*, *magnitude*,
direction components

where *load type label* is TRSHR*n*, TRSHR, TRSHR*n*NU, or TRSHRNU.

**DSLOAD*
surface name, TRSHR or TRSHRNU, *magnitude*, *direction components*

Abaqus/CAE Usage: Use the following input to define an element-based shear surface traction:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction: Shear**, **Distribution**: select an analytical field

Use the following input to define a surface-based general surface traction:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction: Shear**, **Distribution**: **Uniform** or **User-defined**

Nonuniform element-based shear surface traction is not supported in Abaqus/CAE.

Defining the Direction Vector with Respect to a Local Coordinate System

By default, the components of the shear traction vector are specified with respect to the global directions. You can also refer to a local coordinate system (see *Orientations*) for the direction components of these tractions.

Input File Usage: Use one of the following options to specify a local coordinate system:

**DLOAD*, ORIENTATION=*name*
**DSLOAD*, ORIENTATION=*name*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: select **CSYS: Picked** and click **Edit** to pick a local coordinate system, or select **CSYS: User-defined** to enter the name of a user subroutine that defines a local coordinate system

Examples: Using a Local Coordinate System to Define Shear Directions

It is sometimes convenient to give shear and general traction directions with respect to a local coordinate system. The following two examples illustrate the specification of the direction of a shear traction on a cylinder using global coordinates in one case and a local cylindrical coordinate system in the other case. The axis of symmetry

of the cylinder coincides with the global z-axis. A surface named SURFA has been defined on the outside of the cylinder.

In the first example the direction of the shear traction, $\mathbf{t}_{user} = (0., 1., 0.)$, is given in global coordinates. The sense of the resulting shear tractions using global coordinates is shown in *Figure 3*(a).

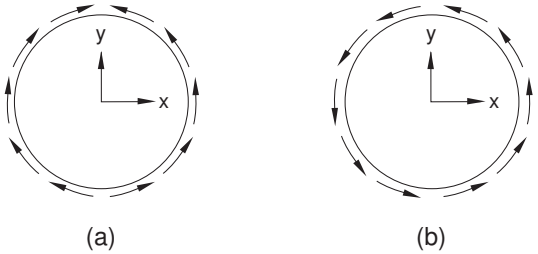


Figure 3: Shear tractions specified using global coordinates (a) and a local cylindrical coordinate system (b).

```
*STEP
 Step 1 - Specify shear directions in global coordinates
...
*DSLOAD
 SURFA, TRSHR, 1., 0., 1., 0.
...
*END STEP
```

In the second example the direction of the shear traction, $\mathbf{t}_{user} = (0., 1., 0.)$, is given with respect to a local cylindrical coordinate system whose axis coincides with the axis of the cylinder. The sense of the resulting shear tractions using the local cylindrical coordinate system is shown in *Figure 3*(b).

```
*ORIENTATION, NAME=CYLIN, SYSTEM=CYLINDRICAL
 0., 0., 0., 0., 0., 1.
...
*STEP
 Step 1 - Specify shear directions in local cylindrical coordinates
...
*DSLOAD, ORIENTATION=CYLIN
 SURFA, TRSHR, 1., 0., 1., 0.
...
*END STEP
```

Resultant Loads due to Surface Tractions

You can choose to integrate surface tractions over the current or the reference configuration by specifying whether or not a constant resultant should be maintained.

In general, the constant resultant method is best suited for cases where the magnitude of the resultant load should not vary with changes in the surface area. However, it is up to you to decide which approach is best for your analysis. An example of an analysis using a constant resultant can be found in *Distributed traction and edge loads*.

Choosing Not to Have a Constant Resultant

If you choose not to have a constant resultant, the traction vector is integrated over the surface in the current configuration, a surface that in general deforms in a geometrically nonlinear analysis. By default, all surface tractions are integrated over the surface in the current configuration.

Input File Usage: Use one of the following options:

*DLOAD, CONSTANT RESULTANT=NO
*DSLOAD, CONSTANT RESULTANT=NO

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction is defined per unit deformed area**

Maintaining a Constant Resultant

If you choose to have a constant resultant, the traction vector is integrated over the surface in the reference configuration and then held constant.

Input File Usage: Use one of the following options:

*DLOAD, CONSTANT RESULTANT=YES
*DSLOAD, CONSTANT RESULTANT=YES

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction is defined per unit undeformed area**

Example

The constant resultant method has certain advantages when a traction is used to model a distributed load with a known constant resultant. Consider the case of modeling a uniform dead load, magnitude p, acting on a flat plate whose normal is in the $\mathbf{e_2}$-direction in a geometrically nonlinear analysis (*Figure 4*).

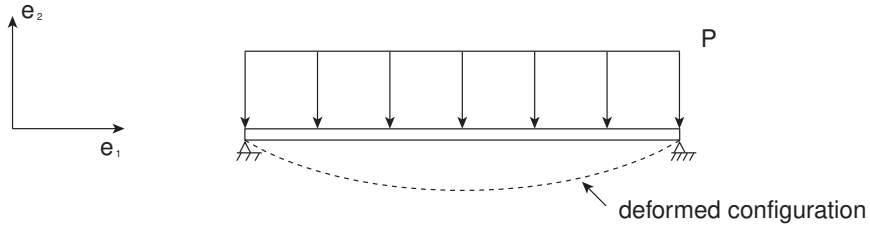


Figure 4: Dead load on a flat plate.

Such a model might be used to simulate a snow load on a flat roof. The snow load could be modeled as a distributed dead traction load $\mathbf{t} = -p\mathbf{e_2}$. Let S_o and S denote the total surface area of the plate in the reference and current configurations, respectively. With no constant resultant, the total integrated load on the plate, $\mathbf{f}$, is

$$\mathbf{f} = \int_S \mathbf{t}\, dS = \int_S -p\,\mathbf{e_2}\, dS = -p\,\mathbf{e_2}\, S.$$

In this case a uniform traction leads to a resultant load that increases as the surface area of the plate increases, which is not consistent with a fixed snow load. With the constant resultant method, the total integrated load on the plate is

$$\mathbf{f} = \int_{S_o} \mathbf{t}\, dS_o = \int_{S_o} -p\,\mathbf{e_2}\, dS_o = -p\,\mathbf{e_2}\, S_o.$$

In this case a uniform traction leads to a resultant that is equal to the pressure times the surface area in the reference configuration, which is more consistent with the problem at hand.

Specifying Pressure Loads

Distributed pressure loads can be specified on any two-dimensional, three-dimensional, or axisymmetric elements. Fluid pressure penetration loads can be specified on any two-dimensional, three-dimensional, or axisymmetric elements. Hydrostatic pressure loads can be specified in Abaqus/Standard on two-dimensional, three-dimensional, and axisymmetric elements. Viscous and stagnation pressure loads can be specified in Abaqus/Explicit on any elements.

Distributed Pressure Loads

Distributed pressure loads can be specified on any elements. For beam elements, a positive applied pressure results in a force vector acting along the particular local direction of the section or a global direction, whichever is specified. For conventional shell elements, the force vector points along the element SPOS normal. For continuum solid or a continuum shell elements with the distributed load on an explicitly identified facet, the force vector acts against the outward normal of that facet. Distributed pressure loads are not supported for pipe and elbow elements.

Distributed pressure loads can be specified on a surface formed over elements; a positive applied pressure results in a force vector acting against the local surface normal.

Input File Usage: Use one of the following options to define a pressure load:

**DLOAD*
element number or element set, *load type label*, *magnitude*

where *load type label* is P*n*, P, P*n*NU, or PNU.

**DSLOAD*
surface name, P or PNU, *magnitude*

Abaqus/CAE Usage: Use the following input to define an element-based pressure load:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pressure** for the **Types for Selected Step**: **Distribution**: select an analytical field or a discrete field

Use the following input to define a surface-based pressure load:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pressure** for the **Types for Selected Step**: **Uniform** or **User-defined**

Nonuniform element-based pressure loads are not supported in Abaqus/CAE.

Fluid Pressure Penetration Loads

You can simulate fluid pressure penetration loads as distributed surface loads or pairwise surface loads. For more information, see *Fluid Pressure Penetration Loads*.

Hydrostatic Pressure Loads on Two-Dimensional, Three-Dimensional, and Axisymmetric Elements in Abaqus/Standard

To define hydrostatic pressure in Abaqus/Standard, give the *Z*-coordinates of the zero pressure level (point *a* in *Figure 5*) and the level at which the hydrostatic pressure is defined (point *b* in *Figure 5*) in an element-based or

surface-based distributed load definition. For levels above the zero pressure level, the hydrostatic pressure is zero.

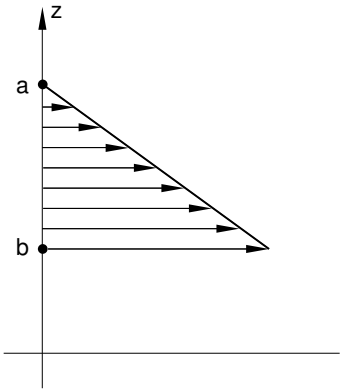


Figure 5: Hydrostatic pressure distribution.

In planar elements the hydrostatic head is in the *Y*-direction; for axisymmetric elements the *Z*-direction is the second coordinate.

Input File Usage: Use one of the following options to define a hydrostatic pressure load:

**DLOAD*
element number or element set, HP*n* or HP, *magnitude*, *Z-coordinate of point a*, *Z-coordinate of point b*

**DSLOAD*
surface name, HP, *magnitude*, *Z-coordinate of point a*, *Z-coordinate of point b*

Abaqus/CAE Usage: Use the following input to define a surface-based hydrostatic pressure load:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pressure** for the **Types for Selected Step**: **Distribution: Hydrostatic**

Element-based hydrostatic pressure loads are not supported in Abaqus/CAE.

Mechanical Pore Pressure Loads on Two-Dimensional, Three-Dimensional, and Axisymmetric Coupled Pore Pressure Elements in Abaqus/Standard

In a coupled pore fluid diffusion and stress analysis (see *Coupled Pore Fluid Diffusion and Stress Analysis*) the pore pressure degrees of freedom, P_{pore}, can be applied automatically as mechanical surface pressures for two-dimensional, three-dimensional, and axisymmetric coupled pore pressure elements. Abaqus/Standard applies a mechanical pressure, $P_{mech} = P_{pore}$, onto the surfaces you prescribe. Since the mechanical pressure loads are determined by the solution pore pressures, Abaqus/Standard ignores any amplitude definition on the distributed load definition. You can include a nonzero scaling factor, α, which will be applied to the pore pressures to give $P_{mech} = \alpha P_{pore}$. By default, the scaling factor is set to unity. This loading is supported only for continuum elements that have pore pressure degrees of freedom during a geostatic (*Geostatic Stress State*) or coupled pore fluid diffusion and stress analysis (*Coupled Pore Fluid Diffusion and Stress Analysis*).

Input File Usage: Use one of the following options to define a mechanical pore pressure load:

**DLOAD*
element number or element set, PORMECH*n*, *scaling factor*

**DSLOAD*
surface name, PORMECH, *scaling factor*

Abaqus/CAE Usage: Mechanical pore pressure loads are not supported in Abaqus/CAE.

Nodal Pressure Loads on Surface Elements for a Multiple Load Case Analysis Involving Substructures

This functionality is limited to distributed loading on surface elements (see *Surface Elements*) in the context of a multiple load case analysis (see *Multiple Linear Load Case Analysis*) involving substructures (see *Generating Substructures*) in Abaqus/Standard. Nodal values of distributed pressure load magnitude are provided via substructure load vectors and can be scaled by a scaling factor associated with this distributed load option. The resulting effect is a continuous pressure field interpolated from nodal values. Abaqus/Standard integrates this pressure field to compute magnitudes of external forces acting at surface nodes in the normal direction.

The surface specified for this type of distributed loading must be based on surface elements and should be connected to the original surface of the structure with surface-based tie constraints (see *Mesh Tie Constraints*). These two surfaces should act as secondary and main surfaces, respectively, in the surface-based tie constraints. These constraints transform the distribution of forces acting on nodes of the surface-element-based surface to a distribution of forces acting on nodes of the original structure.

Input File Usage: Use the following option to define a nodal pressure load:

**DSLOAD*
surface name, NP, *scaling factor*

Abaqus/CAE Usage: Nodal pressure loads are not supported in Abaqus/CAE.

Viscous Pressure Loads in Abaqus/Explicit

Viscous pressure loads are defined by

$$p = -c_v \left(\mathbf{v} - \mathbf{v}_{ref}\right) \cdot \mathbf{n},$$

where p is the pressure applied to the body; c_v is the coefficient of viscosity, given as the magnitude of the load; $\mathbf{v}$ is the velocity of the point on the surface where the pressure is being applied; $\mathbf{v}_{ref}$ is the velocity of the reference node; and $\mathbf{n}$ is the unit outward normal to the element at the same point.

Viscous pressure loading is most commonly applied in structural problems when you want to damp out dynamic effects and, thus, reach static equilibrium in a minimal number of increments. A common example is the determination of springback in a sheet metal product after forming, in which case a viscous pressure would be applied to the faces of shell elements defining the sheet metal. An appropriate choice for the value of c_v is important for using this technique effectively.

To compute c_v, consider the infinite continuum elements described in *Infinite Elements*. In explicit dynamics those elements achieve an infinite boundary condition by applying a viscous normal pressure where the coefficient c_v is given by ρc_d; ρ is the density of the material at the surface, and c_d is the value of the dilatational wave speed in the material (the infinite continuum elements also apply a viscous shear traction). For an isotropic, linear elastic material

$$c_d = \sqrt{\frac{\lambda+2\mu}{\rho}} = \sqrt{\frac{E(1-\nu)}{\rho(1+\nu)(1-2\nu)}},$$

where λ and μ are Lamé's constants, E is Young's modulus, and ν is Poisson's ratio. This choice of the viscous pressure coefficient represents a level of damping in which pressure waves crossing the free surface are absorbed with no reflection of energy back into the interior of the finite element mesh.

For typical structural problems it is not desirable to absorb all the energy (as is the case in the infinite elements). Typically c_v is set equal to a small percentage (perhaps 1 or 2 percent) of ρc_d as an effective way of minimizing ongoing dynamic effects. The c_v coefficient should have a positive value.

Input File Usage: Use one of the following options to define a viscous pressure load:

**DLOAD*, REF NODE=*reference_node*
element number or element set, VP*n* or VP, *magnitude*
**DSLOAD*, REF NODE=*reference_node*
surface name, VP, *magnitude*

Abaqus/CAE Usage: Use the following input to define a surface-based viscous pressure load:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pressure** for the **Types for Selected Step**: **Distribution: Viscous**, toggle on or off **Determine velocity from reference point**

Element-based viscous pressure loads are not supported in Abaqus/CAE.

Stagnation Pressure Loads in Abaqus/Explicit

Stagnation pressure loads are defined by

$$p_s = -c_s(\mathbf{v}\cdot\mathbf{n} - \mathbf{v}_{ref}\cdot\mathbf{n})^2,$$

where p_s is the stagnation pressure applied to the body; c_s is the factor, given as the magnitude of the load; $\mathbf{v}$ is the velocity of the point on the surface where the pressure is being applied; $\mathbf{n}$ is the unit outward normal to the element at the same point; and $\mathbf{v}_{ref}$ is the velocity of the reference node. The coefficient c_s should be very small to avoid excessive damping and a dramatic drop in the stable time increment.

Input File Usage: Use one of the following options to define a stagnation pressure load:

**DLOAD*, REF NODE=*reference_node*
element number or element set, SP*n* or SP, *magnitude*
**DSLOAD*, REF NODE=*reference_node*
element number or element set, SP, *magnitude*

Abaqus/CAE Usage: Use the following input to define a surface-based stagnation pressure load:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pressure** for the **Types for Selected Step**: **Distribution: Stagnation**, toggle on or off **Determine velocity from reference point**

Element-based stagnation pressure loads are not supported in Abaqus/CAE.

Surface Tension loads in Abaqus/Explicit

You can apply surface tension loads as distributed surface loads. For more information, see .

Pressure on Pipe and Elbow Elements

You can specify external pressure, internal pressure, external hydrostatic pressure, or internal hydrostatic pressure on pipe or elbow elements. When pressure loads are applied, the effective outer or inner diameter must be specified in the element-based distributed load definition.

The loads resulting from the pressure on the ends of the element are included: Abaqus assumes a closed-end condition. Closed-end conditions correctly model the loading at pipe intersections, tight bends, corners, and cross-section changes; in straight sections and smooth bends the end loads of adjacent elements cancel each other precisely. If an open-end condition is to be modeled, a compensating point load should be added at the open end. A case where such an end load must be applied occurs if a pressurized pipe is modeled with a mixture of pipe and beam elements. In that case closed-end conditions generate a physically nonexisting force at the transition between pipe and beam elements. Such mixed modeling of a pipe is not recommended.

For pipe elements subjected to pressure loading, the effective axial force due to the pressure loads can be obtained by requesting output variable ESF1 (see *Beam Element Library*).

Input File Usage: Use the following option to define an external pressure load on pipe or elbow elements:

**DLOAD*
element number or element set, PE or PENU, *magnitude*, *effective outer diameter*

Use the following option to define an internal pressure load on pipe or elbow elements:

**DLOAD*
element number or element set, PI or PINU, *magnitude*, *effective inner diameter*

Use the following option to define an external hydrostatic pressure load on pipe or elbow elements:

**DLOAD*
element number or element set, HPE, *magnitude*, *effective outer diameter*

Use the following option to define an internal hydrostatic pressure load on pipe or elbow elements:

**DLOAD*
element number or element set, HPI, *magnitude*, *effective inner diameter*

Abaqus/CAE Usage: Use the following input to define an external or internal pressure load on pipe or elbow elements:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pipe pressure** for the **Types for Selected Step**: **Side**: **External** or **Internal**, **Distribution**: **Uniform**, **User-defined**, or select an analytical field

Use the following input to define an external or internal hydrostatic pressure load on pipe or elbow elements:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pipe pressure** for the **Types for Selected Step**: **Side**: **External** or **Internal**, **Distribution**: **Hydrostatic**

Defining Distributed Surface Loads on Plane Stress Elements

Plane stress theory assumes that the volume of a plane stress element remains constant in a large-strain analysis. When a distributed surface load is applied to an edge of plane stress elements, the current length and orientation of the edge are considered in the load distribution, but the current thickness is not; the original thickness is used.

This limitation can be circumvented only by using three-dimensional elements at the edge so that a change in thickness upon loading is recognized; suitable equation constraints (*Linear Constraint Equations*) would be required to make the in-plane displacements on the two faces of these elements equal. Three-dimensional elements along an edge can be connected to interior shell elements by using a shell-to-solid coupling constraint (see *Shell-to-Solid Coupling* for details).

Edge Tractions and Moments on Shell Elements and Line Loads on Beam Elements

Distributed edge tractions (general, shear, normal, or transverse) and edge moments can be applied to shell elements in Abaqus as element-based or surface-based distributed loads. The units of an edge traction are force per unit length. The units of an edge moment are torque per unit length. References to local coordinate systems are ignored for all edge tractions and moments except general edge tractions.

Distributed line loads can be applied to beam elements in Abaqus as element-based distributed loads. The units of a line load are force per unit length.

The distributed edge and line load types that are available in Abaqus, along with the corresponding load type labels, are listed in *Table 7* and *Table 8*. *About the Element Library* lists the distributed edge and line load types that are available for particular elements and the Abaqus/CAE load support for each load type. For element-based loads applied to shell elements, you must identify the edge of the element upon which the load is prescribed in the load type label (for example, EDLD*n* or EDLD*n*NU).

Follower Edge and Line Loads

By definition, the line of action of a *follower* edge or line load rotates with the edge or line in a geometrically nonlinear analysis. This is in contrast to a *nonfollower* load, which always acts in a fixed global direction.

With the exception of general edge tractions on shell elements and the forces per unit length in the global directions on beam elements, all the edge and line loads listed in *Table 7* and *Table 8* are modeled as follower loads. The normal, shear, and transverse edge loads listed in *Table 7* and *Table 8* act in the normal, shear, and transverse directions, respectively, in the current configuration (see *Figure 6*). The edge moment always acts about the shell edge in the current configuration. The forces per unit length in the local beam directions rotate with the beam elements.

Table 7: Distributed edge load types.

Load description	Load type label for element-based loads	Load type label for surface-based loads
General edge traction	EDLD*n*	EDLD
Normal edge traction	EDNOR*n*	EDNOR
Shear edge traction	EDSHR*n*	EDSHR
Transverse edge traction	EDTRA*n*	EDTRA
Edge moment	EDMOM*n*	EDMOM

Load description	Load type label for element-based loads	Load type label for surface-based loads
Nonuniform general edge traction	EDLD*n*NU	EDLDNU
Nonuniform normal edge traction	EDNOR*n*NU	EDNORNU
Nonuniform shear edge traction	EDSHR*n*NU	EDSHRNU
Nonuniform transverse edge traction	EDTRA*n*NU	EDTRANU
Nonuniform edge moment	EDMOM*n*NU	EDMOMNU
Force per unit length in global *X*-, *Y*-, and *Z*-directions (only for beam elements)	PX, PY, PZ	Not applicable
Nonuniform force per unit length in global *X*-, *Y*-, and *Z*-directions (only for beam elements)	PXNU, PYNU, PZNU	Not applicable
Force per unit length in beam local 1- and 2-directions (only for beam elements)	P1, P2	Not applicable
Nonuniform force per unit length in beam local 1- and 2-directions (only for beam elements)	P1NU, P2NU	Not applicable

Table 8: Distributed edge load types in Abaqus/CAE.

Load description	Abaqus/CAE load type
General edge traction	**Shell edge load**
Normal edge traction	
Shear edge traction	
Transverse edge traction	
Edge moment	
Nonuniform general edge traction	**Shell edge load** (surface-based loads only)
Nonuniform normal edge traction	
Nonuniform shear edge traction	
Nonuniform transverse edge traction	
Nonuniform edge moment	
Force per unit length in global *X*-, *Y*-, and *Z*-directions (only for beam elements)	**Line load**
Nonuniform force per unit length in global *X*-, *Y*-, and *Z*-directions (only for beam elements)	
Force per unit length in beam local 1- and 2-directions (only for beam elements)	
Nonuniform force per unit length in beam local 1- and 2-directions (only for beam elements)	

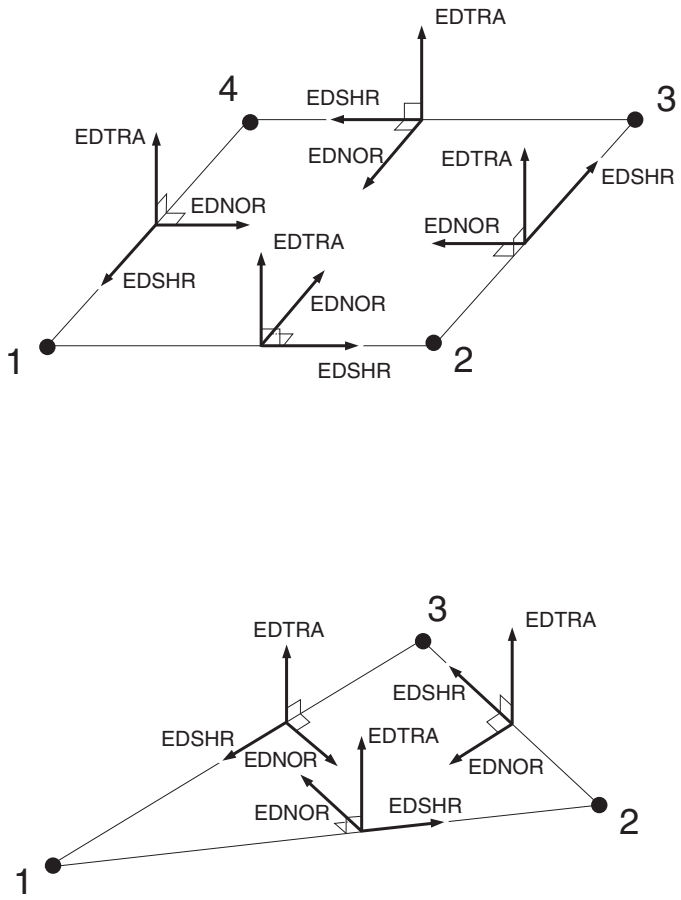


Figure 6: Positive edge loads.

The forces per unit length in the global directions on beam elements are always nonfollower loads.

General edge tractions can be specified to be follower or nonfollower loads. There is no difference between a follower and a nonfollower load in a geometrically linear analysis since the configuration of the body remains fixed.

Input File Usage: Use one of the following options to define general edge tractions as follower loads (the default):

**DLOAD*, FOLLOWER=YES
**DSLOAD*, FOLLOWER=YES

Use one of the following options to define general edge tractions as nonfollower loads:

**DLOAD*, FOLLOWER=NO
**DSLOAD*, FOLLOWER=NO

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction: General**, toggle on or off **Follow rotation**

Specifying General Edge Tractions

General edge tractions allow you to specify an edge load, $\mathbf{t}$, acting on a shell edge, L. The resultant load, $\mathbf{f}$, is computed by integrating $\mathbf{t}$ over L:

$$\mathbf{f} = \int_L \mathbf{t}\, dL.$$

To define a general edge traction, you must provide both a magnitude, α, and direction, $\mathbf{t}_{user}$, for the load. The specified load directions are normalized by Abaqus; thus, they do not contribute to the magnitude of the load.

If a nonuniform general edge traction is specified, the magnitude, α, and direction, $\mathbf{t}_{user}$, must be specified in user subroutine *UTRACLOAD*.

Input File Usage: Use one of the following options to define a general edge traction:

**DLOAD*
element number or element set, EDLD*n* or EDLD*n*NU, *magnitude*, *direction components*
**DSLOAD*
surface name, EDLD or EDLDNU, *magnitude*, *direction components*

Abaqus/CAE Usage: Use the following input to define an element-based general edge traction:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction: General**, **Distribution**: select an analytical field

Use the following input to define a surface-based general edge traction:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction: General**, **Distribution**: **Uniform** or **User-defined**

Nonuniform element-based general edge traction is not supported in Abaqus/CAE.

Rotation of the Load Vector

In a geometrically linear analysis the edge load, $\mathbf{t}$, acts in the fixed direction defined by

$$\mathbf{t} = \alpha \frac{\mathbf{t}_{user}}{\|\mathbf{t}_{user}\|}.$$

If a nonfollower load is specified in a geometrically nonlinear analysis (which includes a perturbation step about a geometrically nonlinear base state), the edge load, $\mathbf{t}$, acts in the fixed direction defined by

$$\mathbf{t} = \alpha \frac{\mathbf{t}_{user}}{\|\mathbf{t}_{user}\|}.$$

If a follower load is specified in a geometrically nonlinear analysis (which includes a perturbation step about a geometrically nonlinear base state), the components must be defined with respect to the reference configuration. The reference edge traction is defined as

$$\mathbf{t}^o = \alpha \frac{\mathbf{t}_{user}}{\|\mathbf{t}_{user}\|}.$$

The applied edge traction, $\mathbf{t}$, is computed by rigidly rotating $\mathbf{t}^o$ onto the current edge.

Defining the Direction Vector with Respect to a Local Coordinate System

By default, the components of the edge traction vector are specified with respect to the global directions. You can also refer to a local coordinate system (see *Orientations*) for the direction components of these tractions.

Input File Usage: Use one of the following options to specify a local coordinate system:

**DLOAD*, ORIENTATION=*name*
**DSLOAD*, ORIENTATION=*name*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: select **CSYS: Picked** and click **Edit** to pick a local coordinate system, or select **CSYS: User-defined** to enter the name of a user subroutine that defines a local coordinate system

Specifying Shear, Normal, and Transverse Edge Tractions

The loading directions of shear, normal, and transverse edge tractions are determined by the underlying elements. A positive shear edge traction acts in the positive direction of the shell edge as determined by the element connectivity. A positive normal edge traction acts in the plane of the shell in the inward direction. A positive transverse edge traction acts in a sense opposite to the facet normal. The directions of positive shear, normal, and transverse edge tractions are shown in *Figure 6*.

To define a shear, normal, or transverse edge traction, you must provide a magnitude, α for the load.

If a nonuniform shear, normal, or transverse edge traction is specified, the magnitude, α, must be specified in user subroutine *UTRACLOAD*.

In a geometrically linear step, the shear, normal, and transverse edge tractions act in the tangential, normal, and transverse directions of the shell, as shown in *Figure 6*. In a geometrically nonlinear analysis the shear, normal, and transverse edge tractions rotate with the shell edge so they always act in the tangential, normal, and transverse directions of the shell, as shown in *Figure 6*.

Input File Usage: Use one of the following options to define a directed edge traction:

**DLOAD*
element number or element set, *directed edge traction label*, *magnitude*
**DSLOAD*
surface name, *directed edge traction label*, *magnitude*

For element-based loads the *directed edge traction label* can be EDSHR*n* or EDSHR*n*NU for shear edge tractions, EDNOR*n* or EDNOR*n*NU for normal edge tractions, or EDTRA*n* or EDTRA*n*NU for transverse edge tractions.

For surface-based loads the *directed edge traction label* can be EDSHR or EDSHRNU for shear edge tractions, EDNOR or EDNORNU for normal edge tractions, or EDTRA or EDTRANU for transverse edge tractions.

Abaqus/CAE Usage: Use the following input to define an element-based directed edge traction:

Load module: **Create Load**; choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**; **Traction: Normal**, **Transverse**, or **Shear**; **Distribution**: select an analytical field

Use the following input to define a surface-based directed edge traction:

Load module: **Create Load**; choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**; **Traction: Normal**, **Transverse**, or **Shear**; **Distribution**: **Uniform** or **User-defined**

Nonuniform element-based directed edge traction is not supported in Abaqus/CAE.

Specifying Edge Moments

An edge moment acts about the shell edge with the positive direction determined by the element connectivity. The directions of positive edge moments are shown in *Figure 7*.

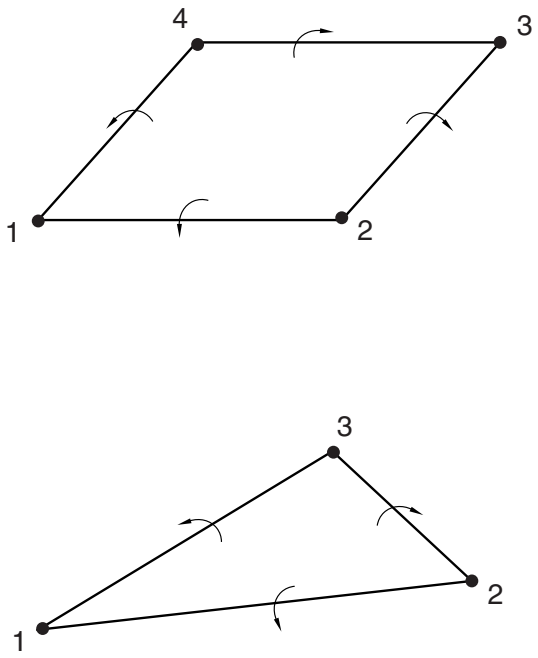


Figure 7: Positive edge moments.

To define a distributed edge moment, you must provide a magnitude, α, for the load.

If a nonuniform edge moment is specified, the magnitude, α, must be specified in user subroutine *UTRACLOAD*.

An edge moment always acts about the current shell edge in both geometrically linear and nonlinear analyses.

In a geometrically linear step an edge moment acts about the shell edge as shown in *Figure 7*. In a geometrically nonlinear analysis an edge moment always acts about the shell edge as shown in *Figure 7*.

Input File Usage: Use one of the following options to define an edge moment:

**DLOAD*
element number or element set, EDMOM*n* or EDMOM*n*NU, *magnitude*
**DSLOAD*
surface name, EDMOM or EDMOMNU, *magnitude*

Abaqus/CAE Usage: Use the following input to define an element-based edge moment:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction: Moment**, **Distribution**: select an analytical field

Use the following input to define a surface-based edge moment:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction: General**, **Distribution**: **Uniform** or **User-defined**

Nonuniform element-based edge moments are not supported in Abaqus/CAE.

Resultant Loads due to Edge Tractions and Moments

You can choose to integrate edge tractions and moments over the current or the reference configuration by specifying whether or not a constant resultant should be maintained. In general, the constant resultant method is best suited for cases where the magnitude of the resultant load should not vary with changes in the edge length. However, it is up to you to decide which approach is best for your analysis.

Choosing Not to Have a Constant Resultant

If you choose not to have a constant resultant, an edge traction or moment is integrated over the edge in the current configuration, an edge whose length changes during a geometrically nonlinear analysis.

Input File Usage: Use one of the following options:

**DLOAD*, CONSTANT RESULTANT=NO
**DSLOAD*, CONSTANT RESULTANT=NO

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction is defined per unit deformed area**

Maintaining a Constant Resultant

If you choose to have a constant resultant, an edge traction or moment is integrated over the edge in the reference configuration, whose length is constant.

Input File Usage: Use one of the following options:

**DLOAD*, CONSTANT RESULTANT=YES
**DSLOAD*, CONSTANT RESULTANT=YES

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction is defined per unit undeformed area**

Specifying Line Loads on Beam Elements

You can specify line loads on beam elements in the global *X*-, *Y*-, or *Z*-direction. In addition, you can specify line loads on beam elements in the beam local 1- or 2-direction.

Input File Usage: Use the following option to define a force per unit length in the global *X*-, *Y*-, or *Z*-direction on beam elements:

**DLOAD*
element number or element set, *load type label*, *magnitude*

where *load type label* is PX, PY, PZ, PXNU, PYNU, or PZNU.

Use the following option to define a force per unit length in the beam local 1- or 2-direction:

**DLOAD*
element number or element set, *load type label*, *magnitude*

where *load type label* is P1, P2, P1NU, or P2NU.

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Line load** for the **Types for Selected Step**

References

- Genta, G., Dynamics of Rotating Systems, Springer , 2005.

Surface Tension Loads

Products: Abaqus/Standard Abaqus/Explicit

References:

- **DSLOAD*
- **SURFACE*
- **SURFACE TENSION*
- **SURFACE TENSION CONTACT ANGLE*
- **CONTACT*

Overview

You can apply surface tension loads as distributed surface loads.

Surface tension loads are prescribed by:

- defining the surface tension coefficients either directly on an element surface or as a surface property; and
- defining the contact angle that is applied only when the surface is in contact with another surface.

Surface Tension

Surface tension is a physical property of liquids that arises from the cohesive forces between liquid molecules at the liquid-vapor interface. The molecules inside the liquid experience balanced intermolecular forces from all directions. However, the molecules at the surface experience a net inward force because they lack neighboring molecules on one side. This imbalance creates a tendency for the liquid surface to minimize its surface area. Surface tension acts like an invisible thin elastic membrane that forms on the surface.

Surface tension represents the force per unit length acting along the surface of a fluid. It plays a critical role at the interfaces between liquids and deformable structures for various scientific, industrial, and medical applications. Practical examples include electronic chip manufacturing (for example, surface tension-driven self-assembly during solder reflow), glass forming, and capillary action. In finite element simulations, surface tension can be modeled as a distributed membrane tension force acting tangentially on the exterior surface of the material. In Abaqus/Standard or Abaqus/Explicit, it can be applied as surface-based distributed loads with load type label ST.

Specifying Surface Tension Loads

Surface tension is an inherent property of the liquid-gas (or liquid-vapor) interface. Therefore, its value depends on the surrounding environment (which usually is not modeled with finite elements).

The effect of surface tension in mechanical systems becomes more significant when the material is a fluid or a soft solid (for example, gel, molten glass, etc.). Surface tension leads to a pressure difference, Δp, between the fluid pressure, p_{int} and the environment pressure, p_{ext}, which is described by the Young-Laplace equation:

$$\Delta p = p_{\text{int}} - p_{ext} = \gamma \nabla_s \cdot n,$$

where γ is the surface tension of the liquid-gas interface (dimensions of force per unit length or energy per unit area), ∇_s is the surface gradient, and n is the vector normal to the surface. The term $\nabla_s \cdot n$ represents twice the local mean curvature of the interface.

You can specify surface tension loads on the following:

- any solid (continuum) stress/displacement element that supports a three-dimensional stress state (except the incompatible mode element C3D8I),
- plane strain and axisymmetric elements,
- membrane elements and regular axisymmetric membrane elements, and
- general surface elements and regular axisymmetric surface elements.

Usually, membrane and surface elements are used only as a skin on the exterior surface of solid elements (although doing so is not required to apply surface tension loads on solid elements).

Surface tension loads are used primarily to model the effects of surface tension on soft or fluid-like materials. Fluid-like materials typically behave as incompressible or nearly incompressible; therefore, selecting appropriate elements is crucial to applying surface tension loads accurately. Adhere to the following guidelines:

- Some traditional Lagrangian elements (such as full integration elements, prism elements, triangular elements, and tetrahedral elements) are not well suited for modeling surface tension due to volumetric locking. These elements often produce overly stiff behavior, which distorts the surface tension effects and prevents accurate force balancing at interfaces. Therefore, you should avoid these elements for surface tension applications and use them only for mesh transitions when necessary or as filler elements in noncritical areas.
- For general three-dimensional applications, you should use C3D8R elements.

You can apply the surface tension loads to a surface. The surface that contains the element and face information is defined as described in *Element-Based Surface Definition* . In Abaqus/Explicit, you can also apply the surface tension loads to an Eulerian material surface. The Eulerian material surface enclosing a particular Eulerian material instance is defined as described in *Eulerian Surface Definition*. In both cases, you must use load type label ST to specify the surface name or Eulerian material surface name.

Input File Usage: Use the following option to define a surface tension load:

**DSLOAD*
surface name or Eulerian material surface name, ST,

The surface tension γ is ignored if it is specified directly as a surface property.

Abaqus/CAE Usage: Surface tension loads are not supported in Abaqus/CAE.

Specifying Temperature-Dependent and Field Variable–Dependent Surface Tension Coefficients

If the surface tension coefficient is a function of temperature and predefined field variables, you must specify the surface tension data under a surface property definition and assign the name of the surface property to the corresponding surface definition. When you apply surface tension loads on this surface, Abaqus automatically uses the surface tension property defined for the surface.

You can use this method even when the surface tension coefficient is a constant (that is, not a function of temperature or predefined field variables).

Input File Usage: Use both of the following options to define a temperature- and field variable–dependent surface tension coefficient:

**SURFACE PROPERTY*, NAME=*SurfPropName*
**SURFACE TENSION*
,temperature,field variables

In this case, the surface tension coefficient defined on the data line under **DSLOAD* is ignored.

Use the following option to assign a named surface tension property to a surface:

**SURFACE*,NAME=*surface name*, PROPERTY=*SurfPropName*

Abaqus/CAE Usage: Defining surface tension by using a surface property is not supported in Abaqus/CAE.

Specifying the Contact Angle

The contact angle is the equilibrium angle formed between the liquid-solid interface and the liquid-vapor interface due to intermolecular forces. The contact angle is a measure of the surface's wettability, describing the tendency of the liquid to spread or bead up. A small contact angle indicates good wettability (hydrophilic); a large contact angle (more than $90°$) indicates poor wettability (hydrophobic). A given system of solid, liquid, and vapor at a given temperature and pressure has a unique equilibrium contact angle (see *Figure 1*).

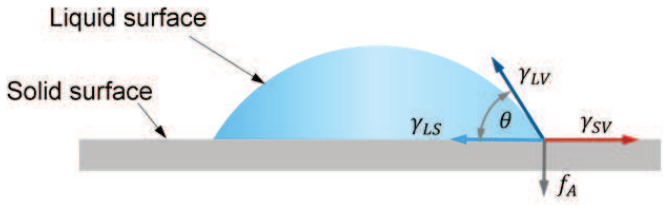


Figure 1: Unique equilibrium contact angle.

Young's equation describes the balance of forces at the point where a liquid droplet meets a solid surface in a vapor:

$$\cos(\theta) = \frac{\gamma_{SV} - \gamma_{LS}}{\gamma_{LV}} \qquad or \qquad \gamma_{LS} - \gamma_{SV} = -\gamma_{LV}\cos(\theta),$$

where θ is the contact angle, γ_{LS} is the solid-liquid interfacial tension, γ_{SV} is the solid-vapor surface tension, and γ_{LV} is the liquid-vapor surface tension.

In typical structural engineering applications, surface tension effects play a significant role on the liquid-vapor and liquid-solid interfaces; usually they can be ignored on the free surfaces of stiff solid components. Using Young's equations, the equilibrium configuration of the system can be achieved by applying a surface tension, γ_{LV}, to the free surface of the liquid (exposed to vapor) and a surface tension of $-\gamma_{LV}\cos(\theta)$ to the surface of the liquid in contact with the solid. This approach requires using free-slip conditions for the interaction between the fluid and the solid (that is, free-slip wall conditions); otherwise, the equilibrium contact angle might not be established properly.

If contact conditions are known a priori, you can directly specify different surface tension values on the free and contacting surfaces of the liquid.

In analyses using Eulerian elements, the contact angle enforcement is inherently approximate. The contact angle cannot generally be satisfied exactly at the wall because the material interface is tracked through volume fractions and level-set functions and it is subject to numerical diffusion. The interface is typically spread over several elements, and the exact location of the interface is not defined uniquely. Mesh refinement can improve the

resolution of the interface and reduces the error. However, the calculated contact angle typically converges toward the prescribed value rather than matching it exactly. Therefore, you should interpret the prescribed contact angle as a target equilibrium angle within the discretization accuracy of the Eulerian formulation.

Input File Usage: Use both of the following options to define the contact angle:

**SURFACE INTERACTION*, NAME=*SurfInterActName*
**SURFACE TENSION CONTACT ANGLE*
(unit degree), *temperature*, *field variables*

Abaqus/CAE Usage: Defining the surface tension contact angle is not supported in Abaqus/CAE.

Assigning a Named Surface Interaction

You must assign a named surface interaction to the contact property for individual surfaces that are used in surface tension loads. The contact angle defined in conjunction with the surface interaction is ignored if it is not assigned to the contact properties.

Input File Usage: Use the following options to assign a named surface interaction to the contact property:

**CONTACT PROPERTY ASSIGNMENT*
, *surface name*, *SurfInterActName*

To activate the contact status, the surface name must be the name used for the surface tension loads.

Abaqus/CAE Usage: Defining surface tension by using a named surface interaction is not supported in Abaqus/CAE.

Examples

Example: Assigning Surface Tension Property to Surface Tension Loads

Use the following options to assign surface tension property to surface tension loads:

```
*SURFACE PROPERTY, NAME=LiquidPropName
*SURFACE TENSION , DEPENDENCIES=1
  3.8E-8, , 100
  4.8E-8, , 200
...
*SURFACE, NAME=LiquidSurf, PROPERTY=LiquidPropName

...
*DSLOAD
LiquidSurf, ST,
```

Example: Modeling the Interaction Directly with Boundary Conditions

This example shows a liquid-solid interaction modeled directly with boundary conditions equivalent to the no-separation, free-slip condition (see *Figure 2*).

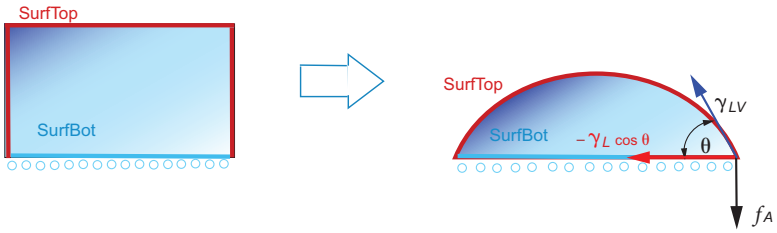


Figure 2: Interaction modeled with boundary conditions.

```
*DSLOAD
surfTop, ST, γLV
surfBot, ST, −γLV cos (θ)
```

Alternatively, you can define the contact angle as part of the surface interaction definition and use the general contact algorithm with the no-separation, free-slip condition to determine the value of the surface tension applied to individual element surfaces based on current contact conditions automatically. In this case Abaqus automatically adjusts the magnitude of the surface tension based on the evolving contact status (see *Figure 3*).



Figure 3: Interaction modeled with no separation, free-slip contact.

Example: Defining the Contact Angle as a Surface Interaction Property

In this example, the contact angle is defined as a surface interaction property. The same contact angle value is defined for all possible contact interactions between the liquid surface and all other surfaces (see *Figure 4*).

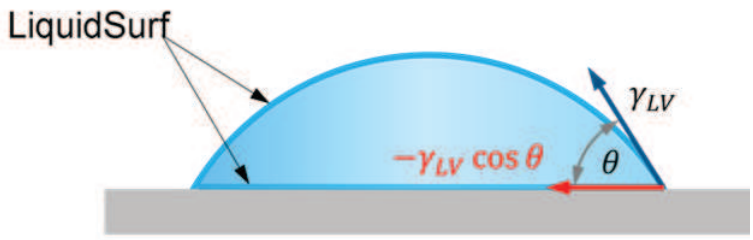


Figure 4: Defining the contact angle as a surface interaction property.

```
*SURFACE, NAME=LiquidSurf, PROPERTY=SurfPropName
*SURFACE PROPERTY, NAME=SurfPropName
*SURFACE TENSION
γ, temperature, field variables

...
*SURFACE INTERACTION, NAME=SurfInterActName
*SURFACE TENSION CONTACT ANGLE
θ, temperature, field variables
...
*CONTACT
*CONTACT PROPERTY ASSIGNMENT
  , LiquidSurf, SurfInterActName
...
*DSLOAD, AMPLITUDE=Amplitude_Name
LiquidSurf, ST,
```

Example: Defining the Contact Angles Using Different Surface Interaction Property Definitions

In this example, pairwise specification of contact angles is defined by using different surface interaction property definitions. The contact angle depends on whether the liquid is in contact with solid material 1 or solid material 2 (see *Figure 5*).

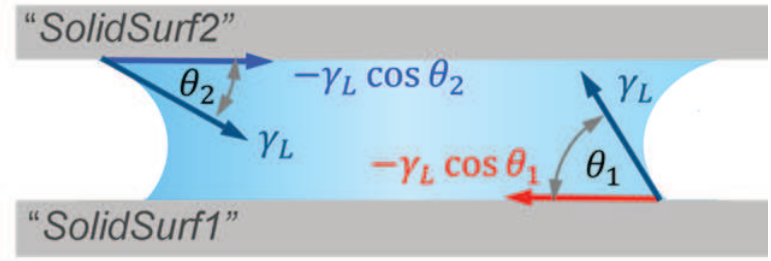


Figure 5: Defining contact angles as the different surface interaction properties.

```
*SURFACE, NAME=LiquidSurf, PROPERTY=SurfPropName
*SURFACE PROPERTY, NAME=SurfPropName
*SURFACE TENSION
γL, temperature, field variables

...
*SURFACE INTERACTION, NAME=SurfInterActName_1
*SURFACE TENSION CONTACT ANGLE
θ1, temperature, field variables
...

*SURFACE INTERACTION, NAME=SurfInterActName_1
*SURFACE TENSION CONTACT ANGLE
θ2, temperature, field variables
...
*CONTACT
*CONTACT PROPERTY ASSIGNMENT
SolidSurf1, LiquidSurf, SurfInterActName_1
SolidSurf2, LiquidSurf, SurfInterActName_2
...
*DSLOAD, AMPLITUDE=Amplitude_Name
LiquidSurf, ST,
```

Contact can be established at any time during the analysis. Abaqus automatically adjusts the magnitude of the surface tension based on the evolving contact status (see *Figure 6*).

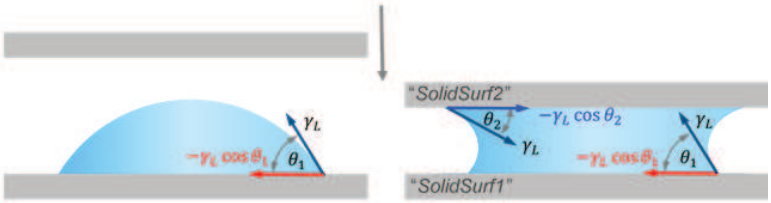


Figure 6: Surface tension affected by contact.

References

- Weatherburn, C. E., Differential Geometry of Three Dimensions, Cambridge University , 1955.
- Saksono, P. H., , and D. Peri , "On Finite Element Modelling of Surface Tension: Variational Formulation and Applications – Part I: Quasistatic Problems," *Computational Mechanics*, vol. 38 265, 2006.

Fluid Pressure Penetration Loads

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- **DSLOAD*
- **PRESSURE PENETRATION*
- **SURFACE*
- **CONTACT*
- **CONTACT PAIR*
- *Defining pressure penetration*

Overview

You can simulate fluid pressure penetration loads as distributed surface loads or pairwise surface loads.

Fluid pressure penetration loads simulated as distributed surface loads:

- model the penetration of fluid on a surface as distributed surface pressure penetration loads that considers the contact pressure field arising from general contact; and
- allow the fluid to penetrate from multiple locations on the surface.

Fluid pressure penetration loads simulated as pairwise surface loads:

- model the penetration of fluid as pairwise pressure penetration loads between specified interfaces with contact pairs; and
- allow the fluid to penetrate from multiple locations on the surface.

Simulating Effects of Fluid Pressure Acting on Surfaces that Consider Contact Conditions

You can model the effects of fluid pressure acting on surfaces without directly modeling the fluid volume. The extent of a surface exposed to fluid pressure can evolve as contact conditions change. Consider the example shown in *Figure 1* and *Figure 2*. A key purpose of the simulation might be to determine if the seal is able to contain the pressurized fluid.

In this type of simulation, one or more initial steps are conducted without the effects of a fluid pressure load to compress the seal. Fluid pressure loading is introduced in a subsequent step. At the beginning of this step, a surface pressure load of magnitude p_f acts on surface regions exposed to the fluid on the left side of *Figure 1*, where the contact pressure is below a specified threshold value, σ_{cpress}^{crit}. The surface pressure load might cause the contact pressure to drop below σ_{cpress}^{crit} over a greater area, leading to more of the surface being exposed to the fluid as shown in *Figure 2*. Various design factors influence whether fluid penetration continues until failure of the seal in these types of studies.

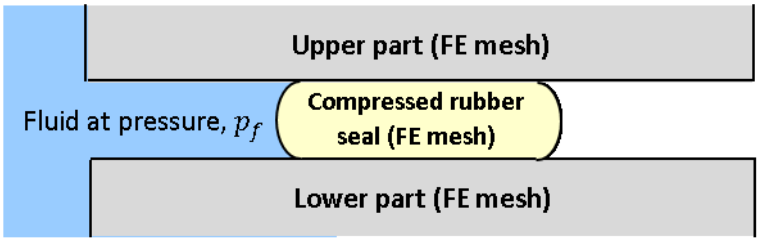


Figure 1: Initial configuration for a fluid pressure loading example.

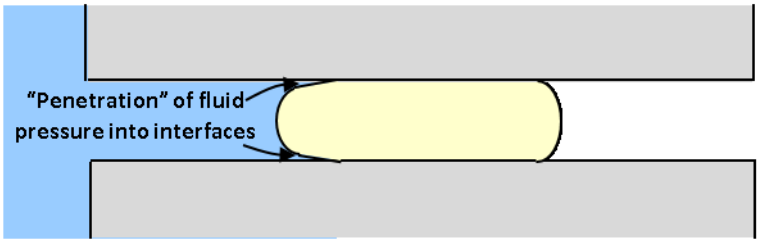


Figure 2: Configuration after some "penetration" of fluid pressure into interfaces.

Abaqus provides two methods for fluid pressure penetration loading that evolves based on contact conditions:

1. Distributed surface pressure penetration loading used in conjunction with general contact and available in both Abaqus/Standard and Abaqus/Explicit.
2. Surface pairwise pressure penetration loading used in conjunction with contact pairs and available only in Abaqus/Standard.

These two methods overlap in functionality and control. Similarities include the ability to control the region initially exposed to the fluid pressure, the magnitude of the fluid pressure, and the critical contact pressure below which fluid penetration starts to occur.

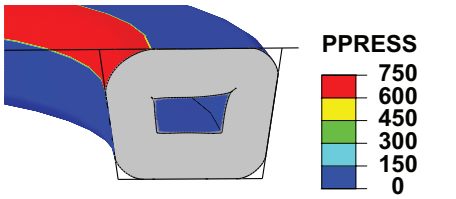
Distinctions between the methods include the approach to defining the contact and loading. Usage details are discussed in *Surface Fluid Pressure Penetration Loading with General Contact* and *Pairwise Fluid Pressure Penetration Loading with Contact Pairs in Abaqus/Standard*.

Results for axisymmetric O-ring seal simulations are shown in *Figure 3* and *Figure 4*. For additional discussion, see *Fluid pressure penetration analysis of an O-ring seal of a pipe connection*. The O-ring is compressed into a cavity between analytical rigid surfaces prior to the introduction of fluid pressure penetration loading. The axisymmetric model of the seal is swept circumferentially to facilitate visualization of results *Figure 3* and *Figure 4*, although the two analytical surfaces are not swept. These simulations consider effects of fluid entering the top, left corner of the cavity to determine if the seal design successfully contains the fluid. *Figure 3* shows contour plots of fluid pressure loading (PPRESS) in the top left corner of the cavity very early in the process of applying fluid pressure penetration loading for three simulation variants modeling the same physics:

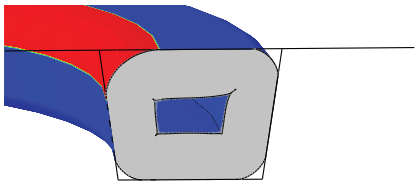
1. An Abaqus/Standard simulation with the distributed surface pressure penetration loading capability in conjunction with general contact.
2. An Abaqus/Explicit simulation with the distributed surface pressure penetration loading capability in conjunction with general contact.
3. An Abaqus/Standard simulation with the surface pairwise pressure penetration loading capability in conjunction with contact pairs.

For this example, the two Abaqus/Standard simulations use the implicit dynamics procedure, and the Abaqus/Explicit simulation uses the explicit dynamics procedure. Contour plots of PPRESS in *Figure 4* show

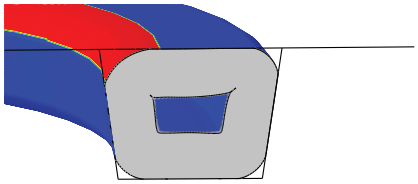
the final extent of the region exposed to fluid pressure for these simulations. All three simulations predict similar distances of fluid penetration into the contact interface.



Surface−based in Abaqus/Standard



Surface−based in Abaqus/Explicit



Pairwise in Abaqus/Standard

Figure 3: Fluid pressure penetration load magnitude (PPRESS) contour plots showing the region initially exposed to fluid pressure for three variants of a simulation.

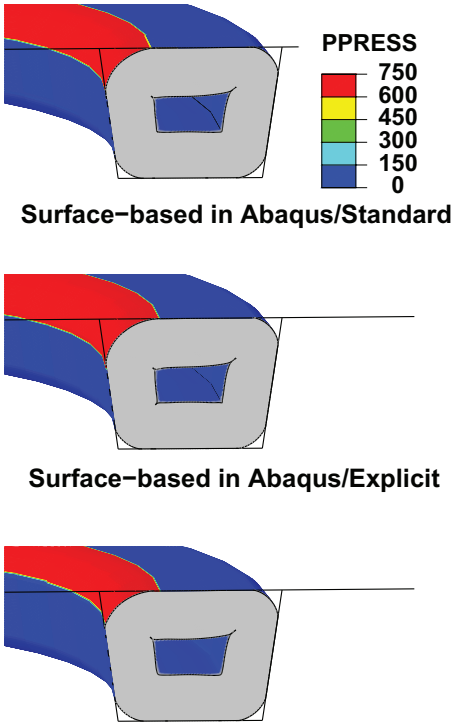


Figure 4: Fluid pressure penetration load magnitude (PPRESS) contour plots showing the final extent of the region exposed to fluid pressure for three variants of a simulation.

Input File Usage:

A potential keyword interface for distributed surface pressure penetration loading in conjunction with general contact for the example shown in *Figure 1* and *Figure 2* is:

```
**Contact specification
*CONTACT
*CONTACT INCLUSIONS, ALL EXTERIOR
…
**Fluid pressure penetration load specification
*DSLOAD
seal_leftside, PPEN, 1.E5, 1.E4, LOCAL
upperpart_leftside, PPEN, 1.E5, 1.E4, LOCAL
lowerpart_leftside, PPEN, 1.E5, 1.E4, LOCAL
```

A potential keyword interface for pairwise pressure penetration loading associated with contact pairs (contact) for the same example is:

```
**Contact specification
*CONTACT PAIR
seal_top, upper_part
seal_bot, lower_part
…
**Fluid pressure penetration load specification
*PRESSURE PENETRATION, SECONDARY=seal_top, MAIN=upper_part
nsetsectop_init, nsetmaintop_init, 1.E5, 1.E4
```

*PRESSURE PENETRATION, SECONDARY=seal_bot, MAIN=lower_part
nsetsecbot_init, nsetmainbot_init, 1.E5, 1.E4

Surface Fluid Pressure Penetration Loading with General Contact

Distributed surface fluid pressure penetration loading simulates the effects of fluid pressure acting on surfaces without directly modeling the fluid. Aspects influencing this loading are discussed in *Magnitude and Time Dependence of the Fluid Pressure* and *Initialization and Evolution of the Region Exposed to Fluid Pressure*.

You specify the surface potentially exposed to fluid pressure loading, the magnitude of the fluid pressure, the critical contact pressure below which fluid penetration starts to occur, the name of the algorithm associated with evolution of the region exposed to fluid pressure, the nodes initially exposed to the fluid, and the penetration time. All surface faces potentially exposed to fluid pressure loading with this capability must be included in the general contact domain (see *Defining the General Contact Domain* in Abaqus/Standard and *Defining the General Contact Domain* in Abaqus/Explicit.

Input File Usage:
*DSLOAD, AMPLITUDE=*name*, OP=MOD (default) or NEW
surface, PPEN, *fluid pressure magnitude*, *critical contact pressure*, LOCAL or WETTING ADVANCE,
node or node set, *penetration time*, *amplitude curve related to critical contact pressure*

Abaqus/CAE Usage:
Load module: **Create Load**: choose **Mechanical** for the **Category** and **Fluid pressure penetration** for the **Types for Selected Step**

Magnitude and Time Dependence of the Fluid Pressure

You define the reference magnitude of the fluid pressure; and, optionally, you define the variation of the fluid pressure during a step by referring to an amplitude curve. Surface-based fluid pressure penetration loading is defined in a similar method to other types of distributed surface loads, including for a surface pressure acting over a nonvarying region of a surface (see *Distributed Loads*). For information about specifying the reference magnitude of the fluid pressure, see *Specifying Pressure Loads*. For information about specifying the time dependence of the fluid pressure, see *Defining Time-Dependent Distributed Loads* and *About Prescribed Conditions*. At each new step, you can modify or completely redefine the fluid pressure penetration loading similar to the way that distributed loads are defined (see *About Loads*).

Initialization and Evolution of the Region Exposed to Fluid Pressure

Abaqus provides the following algorithms associated with the initialization and evolution of the region exposed to fluid pressure:

Local algorithm

- A surface face is “wetted” (exposed to fluid pressure) while a measure of the local contact pressure associated with the face, σ_{local}^{cpress}, is less than or equal to a critical contact pressure, σ_{crit}^{cpress}, and, if cohesive behavior is locally defined, the cohesive damage measure, d_{csdmg}, is greater than 0.95. A newly wetted surface face need not be adjacent to an already wetted surface face with this algorithm.
- A surface face is “unwetted” (not exposed to fluid pressure) while the measure of the local contact pressure associated with the face, σ_{local}^{cpress}, is greater than a critical contact pressure, σ_{crit}^{cpress}, or, if cohesive behavior is locally defined, the cohesive damage measure, d_{csdmg}, is less than or equal to 0.95.

- These criteria are applicable when the fluid pressure penetration load is first introduced and thereafter. You do not specify a node or node set initially exposed to fluid pressure in this case.
- There are no limits to the number of changes in wetting status for a face with this algorithm. For example, a surface face might be initially unwetted, then become wetted, and still later become unwetted again.
- This algorithm is available in Abaqus/Standard and Abaqus/Explicit.

Wetting advance algorithm

- Wetting is irreversible with this algorithm. Once a surface face becomes wetted, it remains wetted as long as pressure penetration loading is in effect.
- You must specify either a single node or a node set consisting of nodes that are initially exposed to fluid pressure. These nodes are not required to be adjacent. If all nodes of a surface face are included in this node set, the face is initially wetted.
- Abaqus determines an initial wetted front, consisting of all nodes in the user-specified node or node set, except those nodes whose adjacent faces are all initially wetted. Abaqus updates the set of wetted-front nodes in subsequent increments as additional faces become wetted.
- An initially unwetted face becomes wetted when any of its adjacent nodes are along the wetted front and a measure of the local contact pressure associated with the face, σ_{local}^{cpress}, is less than or equal to a critical contact pressure, σ_{crit}^{cpress}, and, if cohesive behavior is locally defined, the cohesive damage measure, d_{csdmg}, is greater than 0.95. The wetted front can advance by one face per increment in Abaqus/Explicit with this algorithm.
- This algorithm is available only in Abaqus/Explicit.

Figure 5 shows contour plots of the PPRESS output variable for two Abaqus/Explicit simulations with fluid pressure penetration loading for a simple example consisting of a small block sliding along a larger block. Only two contour levels are shown in the plot: the color red indicates fluid pressure exposure, while the color blue indicates no exposure to the fluid pressure. Nodes of the bottom block at the boundary of the red and blue contour regions near the top block have significant contact forces (and contact pressure) before sliding and are at the "wetted front" before sliding. Faces exposed to fluid pressure contribute fluid pressure forces to adjacent nodes; therefore, nodes on a wetted front receive some force due to fluid pressure. The top block is omitted from some plots in *Figure 5* to observe the fluid pressure exposure on the top surface of the bottom block. In this example the top side of the bottom block and all sides except the top of the top block are potentially exposed to fluid pressure. The four vertical sides of the top block are exposed to fluid pressure throughout the simulation for both types of wetting algorithms. Differences between the local and wetting advance algorithms are especially apparent on the top side of the bottom block. With the wetting advance algorithm, once a face becomes wetted it remains wetted regardless of subsequent contact conditions. However, with the local algorithm, the wetted region corresponds to the region currently without significant contact pressure.

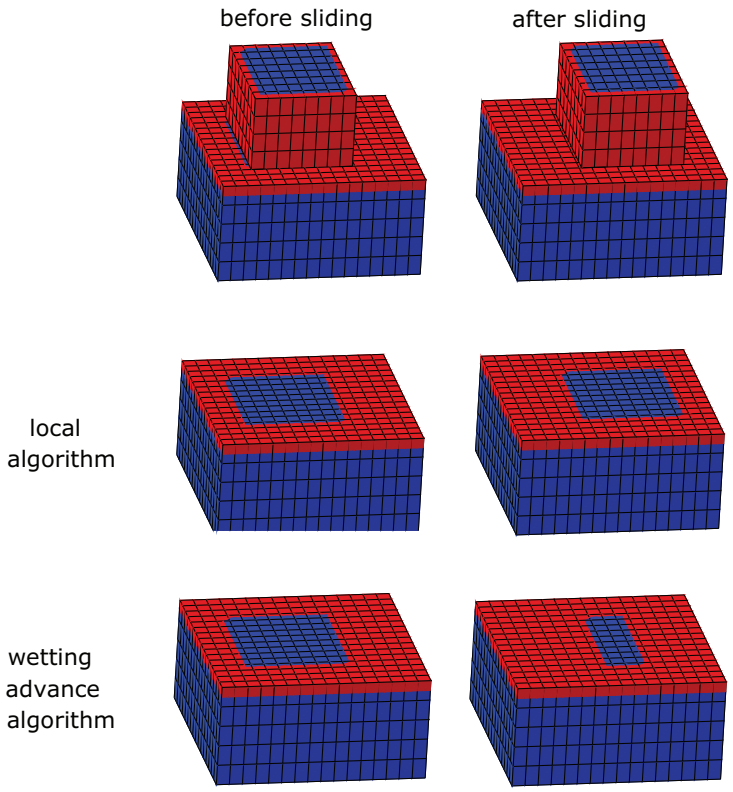


Figure 5: Two-block sliding example with surface-based fluid pressure penetration loading in Abaqus/Explicit.

A critical contact pressure, σ_{cpress}^{crit}, applies to both algorithms. σ_{cpress}^{crit} is zero by default, but it might be appropriate to specify a positive critical contact pressure to account for a tendency for fluid pressure to creep into an interface under low contact pressure due to surface roughness effects. Increasing the value of σ_{cpress}^{crit} tends to increase the surface area exposed to fluid pressure.

The treatment of any overlapping fluid pressure penetration loadings depends on the types of wetting algorithms involved:

- Overlapping fluid pressure penetration loadings involving a mixture of the local and wetting advance algorithms: This is not allowed and triggers an error message.
- Overlapping fluid pressure penetration loadings involving the local algorithm: If a face satisfies the wetting criteria for multiple loadings, the greatest fluid pressure among those loadings is applied, with consideration of current values for time varying fluid pressures.
- Overlapping fluid pressure penetration loadings involving the wetting advance algorithm: Abaqus monitors nodal “fronts” with adjacent faces exposed to different fluid pressure penetration loadings. These fronts can change by one face per increment in Abaqus/Explicit, such that the surface region exposed to the greatest fluid pressure expands.

Magnitude and Time Dependence of the Critical Contact Pressure Threshold

You define the reference magnitude of the critical contact pressure threshold and can optionally define the variation of the critical contact pressure threshold during a step by referring to an amplitude curve. For more information about specifying the critical contact pressure threshold reference magnitude and the name of the amplitude curve, see **DSLOAD*).

Specifying a Penetration Time for the Fluid Pressure in Abaqus/Standard

When the fluid pressure penetration criterion is satisfied, the fluid pressure is applied normal to the surfaces. If the full current fluid pressure is applied immediately, the resulting large changes in the strains near the contact surfaces can cause convergence difficulties. For large-strain problems severe mesh distortion can also occur. To ensure a smooth solution, the fluid pressure is ramped up linearly over a time period from zero pressure penetration load to the full current magnitude.

You can specify the time period taken for the fluid pressure penetration load to reach the full current magnitude on newly penetrated surface segments. If the accumulated increment size, measured immediately after the penetration, is greater than the penetration time, the full current fluid pressure penetration load is applied; otherwise, the fluid pressure on the newly penetrated surface segments is ramped up linearly to the current magnitude over the penetration time period, possibly over a number of increments. When the penetration time is equal to 0, the current fluid pressure is applied immediately once the fluid pressure penetration criterion is satisfied. The default penetration time is chosen to be 0.001 of the total step time. The penetration time is ignored in a linear perturbation step.

Limitations with Surface-based Fluid Pressure Penetration Loads

Fluid pressure penetration loading must be used with a single-sided element-based surface.

It is often good practice, but not required, to define fluid pressure penetration loading on both surfaces of an interface. The loads are applied independently of each other because the loading is surface-based and not pairwise, and the wetted front locations are also determined independently. Thus, the fluid pressure loading can be influenced by how well the mesh densities are matched between the two surfaces along the interface because the wetted front regions along each surface might not match exactly.

While pressure penetration loading acts on a single-sided surface of a shell, the contact pressure field used to compare with the critical contact pressure does not distinguish between contact on either side of the shell. As such, fluid evolution might be inhibited due to contact on the shell side in which the fluid pressure penetration load is not potentially applied.

The fluid pressure penetration load applied at any increment is based on the contact status at the beginning of that increment. Therefore, in Abaqus/Standard, you should be careful in interpreting the results at the end of an increment during which the contact status has changed. Small time increments are recommended to obtain accurate results. This behavior should have minimal effect with Abaqus/Explicit.

In large-displacement Abaqus/Standard analyses, pressure penetration loads introduce unsymmetric load stiffness matrix terms. Using the unsymmetric matrix storage and solution scheme for the analysis step might improve the convergence rate of the equilibrium iterations. For more information on the unsymmetric matrix storage and solution scheme, see *Defining an Analysis*.

The contact pressure field from contact pairs does not have any influence on the evolution of the fluid.

Fluid pressure penetration loading is not available within the following Abaqus/Standard procedures: frequency, complex frequency, buckling, RIKS, substructure generation, and beam section generation. For general procedure types, this loading is not available for steady state transport. However, fluid pressure penetration loads from prior steps can be propagated from the base state of these steps if the steps are linear or remain fixed in the case of a steady state transport step. Fluid pressure penetration loads are not available when you specify a cyclic symmetry mode number because this type of loading does not guarantee cyclic symmetry.

Only solid, shell, axisymmetric, cylindrical, and rigid elements are supported for fluid pressure penetration.

Pairwise Fluid Pressure Penetration Loading with Contact Pairs in Abaqus/Standard

Pairwise fluid pressure penetration loading with contact pairs in Abaqus/Standard is an alternative to surface fluid pressure penetration loading with general contact (see *About Contact Pairs in Abaqus/Standard*). Similarities and differences between these two methods are discussed in *Simulating Effects of Fluid Pressure Acting on Surfaces that Consider Contact Conditions*.

A pairwise fluid pressure penetration loading specification must refer to surfaces of a contact pair. These surface might both be deformable, as is the case with threaded connectors; or one body might be rigid, as occurs when a soft gasket is used as a seal between stiffer structures. Any contact formulation can be used. Other aspects influencing pairwise fluid pressure penetration loading are discussed in *Magnitude and Time Dependence of the Fluid Pressure* and *Initialization and Evolution of the Region Exposed to Fluid Pressure*.

You specify the main and secondary surfaces of the contact pair, nodes exposed to the fluid pressure, the magnitude of the fluid pressure, the critical contact pressure below which fluid penetration starts to occur, and the penetration time.

Input File Usage: **PRESSURE PENETRATION*, SECONDARY=*secondary1*, MAIN=*main1*, PENETRATION TIME
secondary surface node or node set, *main surface node or node set*, *magnitude*, *critical contact pressure*

If a node set is specified, it can contain only one node in two dimensions; in three dimensions it can contain any number of nodes.

Abaqus/CAE Usage: Interaction module:
Create Interaction: **Surface-to-surface contact (Standard)**, **Name:** *contact_interaction_name*; select main and secondary surfaces
Create Interaction: **Pressure penetration**; **Contact interaction:** *contact_interaction_name*, **Region on Main:** select face, edge, or point, **Region on Secondary:** select face, edge, or point, **Critical Contact Pressure:** *critical contact pressure*, **Fluid Pressure:** *magnitude*

Magnitude and Time Dependence of the Fluid Pressure

You must define the reference magnitude of the fluid pressure. You can define the variation of the fluid pressure during a step by referring to an amplitude curve. By default, the reference magnitude is applied immediately at the beginning of the step or ramped up linearly over the step, depending on the amplitude variation assigned to the step (see *Defining an Analysis*).

The pairwise fluid pressure penetration load is applied to the element surface based on the pressure penetration criterion at the beginning of an increment and remains constant over that increment even if the fluid penetrates further during that increment. A nodal integration scheme is used to integrate the distributed fluid pressure penetration load over an element in two dimensions, while in three dimensions Gauss integration scheme is used; the variation of the distributed fluid pressure over an element is determined by the load magnitudes at the element's nodes.

Input File Usage: Use the following option to define the variation of the fluid pressure during a step:

**PRESSURE PENETRATION*, AMPLITUDE=*name*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Pressure penetration**; **Amplitude:** *name*

Removing or Modifying the Pressure Penetration Loads

After pairwise fluid pressure penetration loads are applied to the element surfaces, they are not removed automatically even when contact between the surfaces is reestablished. At each new step the fluid pressure penetration loading, however, can be modified or completely redefined in a manner similar to the way that distributed loads can be defined (see *About Loads*).

Input File Usage: Use the following option to modify the fluid pressure penetration loads that were applied in previous steps:

**PRESSURE PENETRATION*, OP=MOD (default)

In this case the secondary nodes exposed to the fluid pressure must be specified on the data lines. If the main surface is not an analytical rigid surface, the main nodes exposed to the fluid pressure must also be specified on the data lines for planar or axisymmetric models.

Use the following option to remove all fluid pressure penetration loads and, optionally, to specify new fluid pressure penetration loads:

**PRESSURE PENETRATION*, OP=NEW

When OP=NEW is used to remove all fluid pressure penetration loads, no data line is needed. However, when OP=NEW is used to specify new fluid pressure penetration loads, the nodes exposed to the fluid pressure must be specified on the data lines. OP=NEW must be used when defining new exposed nodes. In addition, when OP=NEW is used to respecify a previously defined pressure penetration load, the fluid pressure loading reverts to its last known configuration first, even if the contact status has subsequently changed.

Abaqus/CAE Usage: Use the following option to modify a fluid pressure penetration that was applied in a previous step:

Interaction module: **Interaction Manager**: select interaction, **Edit**

Use the following option to remove a fluid pressure penetration that was applied in a previous step:

Interaction module: **Interaction Manager**: select interaction, **Deactivate**

Initialization and Evolution of the Region Exposed to Fluid Pressure

The algorithm associated with initialization and evolution of the region exposed to the fluid pressure for pairwise fluid pressure penetration loading with contact pairs in Abaqus/Standard is similar to the WETTING ADVANCE algorithm of surface fluid pressure penetration loading with general contact in that the wetting is irreversible and the wetting region incrementally grows along a wetted front.

The fluid can penetrate from either one or multiple locations of the surface. You must identify a node or node set on the secondary surface of the contacting bodies that defines where the surface is exposed to the fluid pressure. In two dimensions if the main surface is not an analytical rigid surface (see *Analytical Rigid Surface Definition*), you must also identify a node or node set on the main surface that defines where the surface is exposed to fluid pressure. These nodes or node sets are always subjected to the pairwise fluid pressure penetration load if they are on the secondary surface, regardless of their contact status. The wetted surface region expands as fluid effectively penetrates into the interface of the contacting bodies from these nodes or nodes sets until a

point is reached where the contact pressure is greater than the specified critical value, σ_{cpress}^{crit}, cutting off further penetration of the fluid.

σ_{cpress}^{crit} is zero by default, but it might be appropriate to specify a positive critical contact pressure to account for a tendency for fluid pressure to creep into an interface under low contact pressure due to surface roughness effects. Increasing the value of σ_{cpress}^{crit} tends to increase surface areas exposed to fluid pressure.

Figure 6 shows results for two Abaqus/Standard simulations with fluid pressure penetration loading for a small block sliding along a larger block. *Figure 5* shows Abaqus/Explicit results for the same example with the LOCAL and WETTING ADVANCE algorithms using general contact. The color red indicates fluid pressure exposure, while the color blue indicates no exposure to the fluid pressure. With pairwise pressure penetration loading, output indicating fluid pressure exposure is available only on the secondary surface. All sides except the top of the smaller, top block and just the top side of the larger, bottom block are potentially exposed to fluid pressure. The four vertical sides of the top block are exposed to fluid pressure throughout the simulation for both pairwise fluid pressure penetration loading and surface-based fluid pressure penetration loading.

Results on the top surface of the bottom block:

- are similar for the pairwise method in Abaqus/Standard (see middle row of plots in *Figure 6*) and the surface-based method with the WETTING ADVANCE algorithm in Abaqus/Explicit (see bottom row of plots in *Figure 5*); although, for example, the region under fluid pressure extends into the active contact region due to the ramp-down characteristic described in *Fluid Pressure Ramp Versus Sharp Cutoff at the Wetted Front*; and
- are similar for the LOCAL algorithm in Abaqus/Standard (see bottom row of plots in *Figure 6*) and Abaqus/Explicit (see middle row of plots in *Figure 5*).
- the wetted front in Abaqus/Standard is at a location between elements (*Figure 6*) compared to nodal locations in Abaqus/Explicit (*Figure 5*).

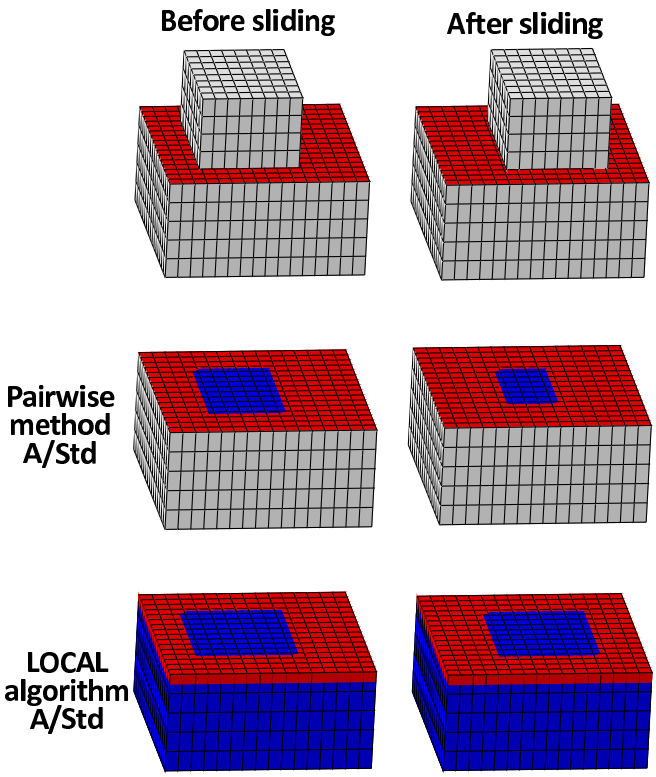


Figure 6: Two-block sliding example comparing pairwise fluid pressure penetration loading and surface-based fluid pressure penetration loading in Abaqus/Standard.

Fluid Pressure Ramp Versus Sharp Cutoff at the Wetted Front

In three dimensions the surfaces of the elements at the front of the penetrated nodes can have only ramped-down pressure loadings. In two dimensions the surfaces of the elements at the front of the penetrated nodes can have either zero or ramped-down pressure loadings.

Input File Usage: Use the following option to apply zero pressure loading to the unwetted surface at the front of the penetration node:

**PRESSURE PENETRATION*, WETTED FRONT=NODE (default)

Use the following option to apply ramped down pressure loading to the unwetted surface at the front of the penetration node:

**PRESSURE PENETRATION*, WETTED FRONT=MID ELEMENT

Abaqus/CAE Usage: Specifying pressure loading on the front unwetted elements is not supported in Abaqus/CAE.

Specifying a Penetration Time for the Fluid Pressure

When the pairwise fluid pressure penetration criterion is satisfied, the fluid pressure is applied normal to the surfaces. If the full current fluid pressure is applied immediately, the resulting large changes in the strains near the contact surfaces can cause convergence difficulties. For large-strain problems severe mesh distortion can

also occur. To ensure a smooth solution, the fluid pressure is ramped up linearly over a time period from zero pressure penetration load to the full current magnitude.

You can specify the time period taken for the pairwise fluid pressure penetration load to reach the full current magnitude on newly penetrated surface segments. If the accumulated increment size, measured immediately after the penetration, is greater than the penetration time, the full current fluid pressure penetration load is applied; otherwise, the fluid pressure on the newly penetrated surface segments is ramped up linearly to the current magnitude over the penetration time period, possibly over a number of increments. When the penetration time is equal to 0, the current fluid pressure is applied immediately once the fluid pressure penetration criterion is satisfied. The default penetration time is chosen to be 0.001 of the total step time. The penetration time is ignored in a linear perturbation step.

Input File Usage: **PRESSURE PENETRATION*, PENETRATION TIME=*n*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Pressure penetration**; **Penetration time:** *n*

Limitations with Pairwise Fluid Pressure Penetration Loads

Each secondary surface subjected to pressure penetration loading must be continuous and cannot be a closed loop. Pressure penetration loading cannot be used with a node-based secondary surface. The pressure penetration load applied at any increment is based on the contact status at the beginning of that increment. You should, therefore, be careful in interpreting the results at the end of an increment during which the contact status has changed. Small time increments are recommended to obtain accurate results.

When pressure penetrates into contacting bodies between an analytical rigid surface and a deformable surface, no pressure penetration load are applied to the analytical rigid surface. The reference node on the analytical rigid surface should, therefore, be constrained in all directions. To account for the effect of fluid pressure penetration loads on the rigid surface, the analytical rigid surface should be replaced with an element-based rigid surface.

When fluid with different pressure loads penetrates into an element simultaneously from multiple locations on a surface, the maximum value of the fluid pressure loads is applied to the element.

In large-displacement analyses pressure penetration loads introduce unsymmetric load stiffness matrix terms. Using the unsymmetric matrix storage and solution scheme for the analysis step might improve the convergence rate of the equilibrium iterations. See *Defining an Analysis* for more information on the unsymmetric matrix storage and solution scheme.

Only solid, shell, cylindrical, and rigid elements are supported for three-dimensional pressure penetration.

Behavior in Linear Perturbation Steps

Perturbation analyses can be performed during a fully nonlinear analysis by including linear perturbation steps between the general analysis steps. With the exception of the static LCP perturbation procedure, contact conditions are not allowed to change during a linear perturbation step; the fluid penetrates no further into the surface and remains as it was defined in the base state. Even in the case of a static LCP perturbation procedure, where the contact status at the end of the perturbation analysis can be different from the base state, the portions of the contact surfaces where the fluid pressure acts remain frozen at the base state. The fluid pressure magnitude applied in the previous general analysis step, however, can be modified during a linear perturbation analysis step. In matrix generation (see *Generating Matrices as a Linear Analysis Step*) and steady-state dynamic analyses (direct or modal—see *Direct-Solution Steady-State Dynamic Analysis* and *Mode-Based Steady-State Dynamic Analysis*) you can specify both the real (in-phase) and imaginary (out-of-phase) parts of the loading.

Input File Usage: Use the following option to define the real (in-phase) part of the surface-based fluid pressure penetration loading:

**DSLOAD*, REAL (default)

Use the following option to define the imaginary (out-of-phase) part of the surface-based fluid pressure penetration loading:

**DSLOAD*, IMAGINARY

Use the following option to define the real (in-phase) part of the pairwise fluid pressure penetration loading:

**PRESSURE PENETRATION*, REAL (default)

Use the following option to define the imaginary (out-of-phase) part of the pairwise fluid pressure penetration loading:

**PRESSURE PENETRATION*, IMAGINARY

The REAL or IMAGINARY parameters are ignored in all procedures other than steady-state dynamics.

Abaqus/CAE Usage: Use the following option to define the real (in-phase) part of the pairwise fluid pressure penetration loading:

Interaction module: **Create Interaction**: **Pressure penetration**; **Fluid Pressure (Real)**

Use the following option to define the imaginary (out-of-phase) part of the pairwise fluid pressure penetration loading:

Interaction module: **Create Interaction**: **Pressure penetration**; **Fluid Pressure (Imaginary)**

Output

You can request output of the currently active pressure (PPRESS) and force (PFORCE) associated with nodes of a wetted region due to fluid pressure penetration loading as surface output to the data, results, and output database files (see *Surface Output from Abaqus/Standard* and *Writing Surface Output to the Output Database*). With pairwise fluid pressure loading, you can define this output on nodes along the secondary surface.

You can also request the total force (PFN) associated with a whole surface due to fluid pressure penetration loading.

Thermal Loads

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Loads*
- **CFLUX*
- **DFLUX*
- **DSFLUX*
- **CFILM*
- **FILM*
- **SFILM*
- **FILM PROPERTY*
- **CRADIATE*
- **RADIATE*
- **SRADIATE*
- *Defining a concentrated heat flux*
- *Defining a body heat flux*
- *Defining a surface heat flux*
- *Defining a surface film condition interaction*
- *Defining a concentrated film condition interaction*
- *Defining a surface radiative interaction*
- *Defining a concentrated radiative interaction*
- *Progressive Element Activation*
- **ELEMENT PROGRESSIVE ACTIVATION*

Overview

You can apply thermal loads in all procedures that solve for nodal temperature degrees of freedom to model conduction, convection, and radiation.

The following types of thermal loads are available:

- Concentrated heat flux prescribed at nodes.
- Distributed heat flux prescribed on element faces or surfaces.
- Body heat flux per unit volume.
- Boundary convection defined at nodes, on element faces, or on surfaces.
- Boundary radiation defined at nodes, on element faces, or on surfaces.
- Moving or stationary concentrated heat fluxes defined in user subroutine *UMDFLUX*.

See *About Loads* for general information that applies to all types of loading.

Modeling Thermal Radiation

The following types of radiation heat exchange can be modeled using Abaqus:

- Exchange between a nonconcave surface and a nonreflecting environment. This type of radiation is modeled using boundary radiation loads defined at nodes, on element faces, or on surfaces, as described below.
- Exchange between two surfaces within close proximity of each other in which temperature gradients along the surfaces are not large. This type of radiation is modeled using the gap radiation capability described in *Thermal Contact Properties*.
- Exchange between surfaces that constitute a cavity. This type of radiation is modeled using the cavity radiation capability available in Abaqus/Standard and described in *Cavity Radiation in Abaqus/Standard* or through the average-temperature radiation condition described in *Specifying Average-Temperature Radiation Conditions* below.

Prescribing Heat Fluxes Directly

Concentrated heat fluxes can be prescribed at nodes (or node sets). Distributed heat fluxes can be defined on element faces or surfaces.

Specifying Concentrated Heat Fluxes

By default, a concentrated heat flux is applied to degree of freedom 11. For shell heat transfer elements concentrated heat fluxes can be prescribed through the thickness of the shell by specifying degree of freedom 11, 12, 13, etc. Temperature variation through the thickness of shell elements is described in *Choosing a Shell Element*.

Input File Usage: **CFLUX*
node number or node set name, *degree of freedom*, *heat flux magnitude*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Thermal** for the **Category** and **Concentrated heat flux** for the **Types for Selected Step**: select region: **Magnitude**: *heat flux magnitude*

Use the following input to define an ion concentration flux (dof=33):

Load module: **Create Load**: choose **Coupled Thermal Electro-Chemical** for the **Category** and **Ion concentration flux** for the **Types for Selected Step**: select region: **Magnitude**: *ion flux magnitude*

Specifying Concentrated Heat Fluxes at Phantom Nodes for Enriched Elements

For an enriched element (see *Modeling Discontinuities as an Enriched Feature Using the Extended Finite Element Method*), you can apply concentrated heat flux at a phantom node that is originally located coincident with the specified real node.

Alternatively, you can apply concentrated heat flux at a phantom node located at an element edge between two specified real corner nodes. This setting applies only to nodes with both pore pressure and temperature degrees of freedom.

Input File Usage: Use the following option to specify concentrated heat fluxes at a phantom node originally located coincident with the specified real node:

**CFLUX*, PHANTOM=NODE
node number, *degree of freedom*, *heat flux magnitude*

Use the following option to specify concentrated heat fluxes at a phantom node located at an element edge:

**CFLUX*, PHANTOM=EDGE
first corner node number, *second corner node number*, *heat flux magnitude*

Abaqus/CAE Usage: Specifying concentrated heat fluxes at phantom nodes for enriched elements is not supported in Abaqus/CAE.

Defining the Values of Concentrated Nodal Flux from a User-Specified File

You can define nodal flux using nodal flux output from a particular step and increment in the output database (`.odb`) file of a previous Abaqus analysis. The part (`.prt`) file from the original analysis is also required when reading data from the output database file. In this case both the previous model and the current model must be defined consistently, including node numbering, which must be the same in both models. If the models are defined in terms of an assembly of part instances, part instance naming must be the same.

Input File Usage: **CFLUX*, FILE=file, STEP=step, INC=inc

Abaqus/CAE Usage: Defining the values of concentrated nodal flux from a user-specified file is not supported in Abaqus/CAE.

Specifying Element-Based Distributed Heat Fluxes

You can specify element-based distributed surface fluxes (on element faces) or body fluxes (flux per unit volume). For surface fluxes you must identify the face of the element on which the flux is prescribed in the flux label (for example, S*n* or S*n*NU for continuum elements). The distributed flux types available depend on the element type. *About the Element Library* lists the distributed fluxes that are available for particular elements.

Input File Usage: **DFLUX*
element number or element set name, *load type label*, *flux magnitude*

where *load type label* is S*n*, SPOS, SNEG, BFCE, or BF

Abaqus/CAE Usage: Use the following input to define a distributed surface flux:

Load module: **Create Load**: choose **Thermal** for the **Category** and **Surface heat flux** for the **Types for Selected Step**: select region: **Distribution**: select an analytical field, **Magnitude**: *flux magnitude*

Use the following input to define a distributed body flux:

Load module: **Create Load**: choose **Thermal** for the **Category** and **Body heat flux** for the **Types for Selected Step**: select region: **Distribution**: **Uniform** or select an analytical field, **Magnitude**: *flux magnitude*

Use the following input to define a distributed ion concentration body flux:

Load module: **Create Load**: choose **Coupled Thermal Electro-Chemical** for the **Category** and **Distributed ion concentration body flux** for the **Types for Selected Step**: select region: **Distribution**: select an analytical field, **Magnitude**: *flux magnitude*

Specifying Surface-Based Distributed Heat Fluxes

When you specify distributed surface fluxes on a surface, the surface that contains the element and face information is defined as described in *Element-Based Surface Definition*. You must specify the surface name, the heat flux label, and the heat flux magnitude.

Input File Usage:
**DSFLUX*
surface name, S, *flux magnitude*

Abaqus/CAE Usage: Use the following input to specify surface-based distributed heat fluxes:

Load module: **Create Load**: choose **Thermal** for the **Category** and **Surface heat flux** for the **Types for Selected Step**: select region: **Distribution**: **Uniform**, **Magnitude**: *flux magnitude*

Modifying or Removing Heat Fluxes

Heat fluxes can be added, modified, or removed as described in *About Loads*.

Specifying Time-Dependent Heat Fluxes

The magnitude of a concentrated or a distributed heat flux can be controlled by referring to an amplitude curve. If different magnitude variations are needed for different fluxes, the flux definitions can be repeated, with each referring to its own amplitude curve. See *About Prescribed Conditions* and *Amplitude Curves* for details.

Defining Nonuniform Distributed Heat Flux in a User Subroutine

A nonuniform element-based or surface-based distributed flux can be defined in Abaqus/Standard and Abaqus/Explicit by using user subroutines *DFLUX* and *VDFLUX*, respectively. In Abaqus/Standard the specified reference magnitude is passed into user subroutine *DFLUX* as `FLUX(1)` (see *DFLUX*). If you omit the magnitude, `FLUX(1)` is passed in as zero. In Abaqus/Explicit the specified reference magnitude that you must define is the variable `VALUE` (see *VDFLUX*).

Input File Usage: Use the following option to define a nonuniform element-based heat flux:

**DFLUX*
element number or element set name, *load type label*

where *load type label* is S*n*NU, SPOSNU, SNEGNU, or BFNU.

Use the following option to define a nonuniform surface-based heat flux:

**DSFLUX*
surface name, SNU

Abaqus/CAE Usage: Use the following input to define a nonuniform element-based body flux:

Load module: **Create Load**: choose **Thermal** for the **Category** and **Body heat flux** for the **Types for Selected Step**: select region: **Distribution: User-defined**, **Magnitude**: *flux magnitude*

Use the following input to define a nonuniform surface-based heat flux:

Load module: **Create Load**: choose **Thermal** for the **Category** and **Surface heat flux** for the **Types for Selected Step**: select region: **Distribution: User-defined**, **Magnitude**: *flux magnitude*

Nonuniform element-based distributed surface fluxes are not supported in Abaqus/CAE.

Defining Moving or Stationary Nonuniform Heat Flux in User Subroutine `UMDFLUX`

Multiple nonuniform concentrated heat fluxes can be defined in user subroutine *UMDFLUX* in Abaqus/Standard. These heat fluxes can be stationary or moving between start points and end points inside the element.

Input File Usage: Use the following option to define nonuniform moving concentrated heat fluxes:

**DFLUX*
element set name, MBFNU, blank entry, *table collection name or leave blank if no table collection is used*

Abaqus/CAE Usage: Defining moving or stationary nonuniform heat flux in user subroutine *UMDFLUX* is not supported in Abaqus/CAE.

Prescribing Boundary Convection

Heat flux on a surface due to convection is governed by

$$q = -h\left(\theta - \theta^0\right),$$

where

q

is the heat flux across the surface,

h

is the film coefficient,

θ

is the temperature at this point on the surface, and

θ^0

is a reference sink temperature value.

You can define heat flux due to convection on element faces, on surfaces, or at nodes.

In the presence of a user-defined film amplitude, A_{film}, and sink amplitude, A_{sink}, the heat flux due to convection is given as:

$$q = -A_{film}h\left(\theta - A_{sink}\theta^0\right),$$

and Abaqus computes the effective sink temperature written to the output database as $A_{sink}\theta^0$.

In the absence of a user-defined film amplitude, Abaqus ramps down the heat flux that is present at the start of the analysis (or at the end of the previous step in a multistep analysis) using the default step amplitude. This enables a smooth transition between steps and improves solver convergence. Here, the heat flux is:

$$q = -h\left(\theta - A_{sink}\theta^0\right) + \left(1 - A_{step}\right)q_0,$$

where

q_0

is the initial heat flux at the start of a step and computed as $q_0 = -h_0\left(\theta_{ini} - {\theta^0}_{ini}\right)$,

h_0

is the film coefficient at the end of the previous step or at the start of the analysis,

A_{step}

is the default amplitude for a procedure that can either change linearly with time throughout the step (ramp function) or is applied immediately and remains constant throughout the step (step function)—see *Defining Time Varying Prescribed Conditions*,

${\theta^0}_{ini}$

is the sink temperature at the end of the previous step or at the start of the analysis, and

θ_{ini}

is the temperature at the end of the previous step or at the start of the analysis.

The effective sink temperature written to the output database is $A_{sink}\theta^0 + \left(1 - A_{step}\right)\frac{q_0}{h_0}$

Specifying Element-Based Film Conditions

You can define the sink temperature value, θ^0, and the film coefficient, h, on element faces. The convection is applied to element edges in two dimensions and to element faces in three dimensions. The edge or face of the element on which the film is placed is identified by a film load type label and depends on the element type (see *About the Element Library*). You must specify the element number or element set name, the film load type label, a sink temperature, and a film coefficient.

Input File Usage: **FILM*

element number or element set name, *film load type label*, θ^0, h

Abaqus/CAE Usage: Element-based film conditions are supported in Abaqus/CAE only for the film coefficient.

Interaction module: **Create Interaction**: **Surface film condition**: select region: **Definition**: select an analytical field: **Film coefficient:** h

Specifying Element-Based Film Conditions on Evolving Faces of an Element in Abaqus/Standard

You can define the sink temperature value, θ^0, and the film coefficient, h, on three-dimensional continuum elements that support the temperature degree of freedom. The convection is applied to element faces in three dimensions. The face of the element on which the film is to be placed is identified automatically at the start of an increment. When elements are added or removed using model change during an analysis or using element activation or element deletion during an increment of a step, the film convection is applied automatically at the start of an increment on the new exposed faces and removed from the unexposed faces. You must specify the element number or element set name, the film load type label, a sink temperature, and a film coefficient.

By default, convection is applied on the exposed full element facet area. When you use partial element activation (see *Progressive Element Activation*), you can use user subroutine *UEPACTIVATIONFACET* to modify the exposed area over which convection is applied. For example, *Figure 1* displays the area fractions of the partially filled facets C-I1-I4, C-B-I2-I1, and B-I3-I2 when partial activation is used. Partial element activation exposes an internal cut surface area represented as I1-I2-I3-I4. You can use user subroutine *UEPACTIVATIONFACET* to specify the convection area on this cut surface. In addition, you can use user subroutine *FILM* to specify different film coefficients for the internal cut surface versus the element facets.

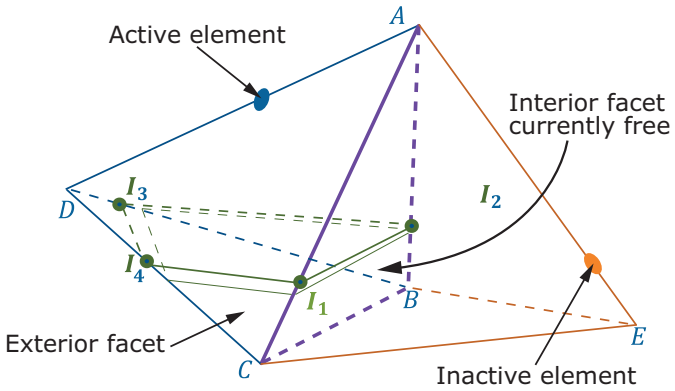


Figure 1: Partial facets and internal free surface for film cooling.

Input File Usage: **FILM*

element number or element set name, FFS or FFSNU, θ^0, h

Abaqus/CAE Usage: Element-based film conditions on evolving faces of element are supported in Abaqus/CAE only for the film coefficient.

Interaction module: **Create Interaction**: **Surface film condition**: select region: **Definition**: select an analytical field: **Film coefficient:** h

Select **Toggle on dynamic surface activation**.

Specifying Surface-Based Film Conditions

You can define the sink temperature value, θ^0, and the film coefficient, h, on a surface. The surface that contains the element and face information is defined as described in *Element-Based Surface Definition*. You must specify the surface name, the film load type, a sink temperature, and a film coefficient.

By default, the specified film coefficient is set to zero on portions of surfaces with active contact (closed status). When the contact status changes with time (such as in a thermomechanical analysis), Abaqus uses the contact status distribution from the previous increment to determine portions of active contact. To accommodate a changing film coefficient due to contact status changes and to ensure robustness, Abaqus/Standard ramps the change to the film coefficient over a time period that is a small fraction of the current step duration. The film coefficient change occurs during one increment in Abaqus/Explicit. Optionally, you can turn off the dependence of the film coefficient on the contact status. If there is no contact in the model, Abaqus ignores the contact status dependence.

Input File Usage: Use the following option to define surface-based film conditions with contact status dependence explicitly (default):

**SFILM*, TRANSITION=TOUCHING

surface name, F or FNU, θ^0, h, time period over which a change to film coefficient is ramped

Use the following option to turn off contact status dependence for surface-based film conditions:

**SFILM*, TRANSITION=NONE

surface name, F or FNU, θ^0, h

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface film condition**: select region: **Definition**: **Embedded Coefficient** or **User-defined**: **Film coefficient:** h and **Sink temperature:** θ^0

Specifying Node-Based Film Conditions

A node-based film condition requires that you define the nodal area for a specified node number or node set; the sink temperature value, θ^0; and the film coefficient, h. The associated degree of freedom is 11. For shell type elements where the film is associated with a degree of freedom other than 11, you can specify the concentrated film for a duplicate node that is constrained to the appropriate degree of freedom of the shell node by using an equation constraint (see *Linear Constraint Equations*).

Input File Usage: **CFILM*

node number or node set name, *nodal area*, θ^0, h

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Concentrated film condition**: select region: **Definition**: **Embedded Coefficient**, **User-defined**, or select an analytical field: **Associated nodal area:** *nodal area*, **Film coefficient:** h, **Sink temperature:** θ^0

Specifying Node-Based Film Conditions at Phantom Nodes for Enriched Elements

For an enriched element (see *Modeling Discontinuities as an Enriched Feature Using the Extended Finite Element Method*), you can define the nodal area; the sink temperature value, θ^0; and the film coefficient, h, at a phantom node that is originally located coincident with the specified real node.

Alternatively, you can define the nodal area; the sink temperature value, θ^0; and the film coefficient, h, at a phantom node located at an element edge between two specified real corner nodes. This setting applies only to nodes with both pore pressure and temperature degrees of freedom.

Input File Usage: Use the following option to define the film conditions at a phantom node originally located coincident with the specified real node:

**CFILM*, PHANTOM=NODE

node number, *nodal area*, θ^0, h

Use the following option to define the film conditions at a phantom node located at an element edge:

**CFILM*, PHANTOM=EDGE

first corner node number, *second corner node number*, *nodal area*, θ^0, h

Abaqus/CAE Usage: Defining film conditions at phantom nodes for enriched elements is not supported in Abaqus/CAE.

Specifying Temperature- and Field-Variable-Dependent Film Conditions

If the film coefficient is a function of temperature, you can specify the film property data separately and specify the name of the property table instead of the film coefficient in the film condition definition.

You can specify multiple film property tables to define different variations of the film coefficient, h, as a function of surface temperature and/or field variables. Each film property table must be named. This name is referred to by the film condition definitions.

A new film property table can be defined in a restart step. If a film property table with an existing name is encountered, the second definition is ignored.

Input File Usage: For element-based film conditions, use the following options:

**FILM PROPERTY*, NAME=*film property table name*
**FILM*

element number or element set name, *film load type label*, θ^0, *film property table name*

For surface-based film conditions, use the following options:

**FILM PROPERTY*, NAME=*film property table name*
**SFILM*

surface name, F, θ^0, *film property table name*

For node-based film conditions, use the following options:

**FILM PROPERTY*, NAME=*film property table name*
**CFILM*

node number or node set name, *nodal area*, θ^0, *film property table name*

The **FILM PROPERTY* option must appear in the model definition portion of the input file.

Abaqus/CAE Usage: Interaction module:
Create Interaction Property: **Name**: *film property table name* and **Film condition**
Create Interaction: **Surface film condition** or **Concentrated film condition**: select region: **Definition: Property Reference** and **Film interaction property**: *film property table name*

Modifying or Removing Film Conditions

Film conditions can be added, modified, or removed as described in *About Loads*.

Specifying Time-Dependent Film Conditions

For a uniform film both the sink temperature and the film coefficient can be varied with time by referring to amplitude definitions. One amplitude curve defines the variation of the sink temperature, θ^0, with time. Another amplitude curve defines the variation of the film coefficient, h, with time. See *About Prescribed Conditions* and *Amplitude Curves* for more information.

Input File Usage: Use the following options to define time-dependent film conditions:

**AMPLITUDE*, NAME=*temp_amp*
**AMPLITUDE*, NAME=*h_amp*
**FILM*, SINK AMPLITUDE=*temp_amp*, FILM AMPLITUDE=*h_amp*
**SFILM*, SINK AMPLITUDE=*temp_amp*, FILM AMPLITUDE=*h_amp*
**CFILM*, SINK AMPLITUDE=*temp_amp*, FILM AMPLITUDE=*h_amp*

Abaqus/CAE Usage: Use the following input to define time-dependent film conditions. If you select an analytical field to define the interaction, the analytical field affects only the film coefficient.

Interaction module:
Create Amplitude: **Name:** *h_amp*
Create Amplitude: **Name:** *temp_amp*
Create Interaction: **Surface film condition** or **Concentrated film condition**: select region: **Definition**: **Embedded Coefficient** or select an analytical field: **Film coefficient amplitude**: *h_amp* and **Sink amplitude**: *temp_amp*

Examples

A uniform, time-dependent film condition can be defined for face 2 of element 3 by

```
*AMPLITUDE, NAME=sink
 0.0, 0.5, 1.0, 0.9
*AMPLITUDE, NAME=famp
 0.0, 1.0, 1.0, 22.0
 …
*STEP
** For an Abaqus/Standard analysis:
*HEAT TRANSFER
** For an Abaqus/Explicit analysis:
*DYNAMIC TEMPERATURE-DISPLACEMENT, EXPLICIT
 …
```

```
*FILM, SINK AMPLITUDE=sink, FILM AMPLITUDE=famp
 3, F2, 90.0, 2.0
```

A uniform, temperature-dependent film coefficient and a time-dependent sink temperature can be defined for face 2 of element 3 by

```
*AMPLITUDE, NAME=sink
0.0, 0.5, 1.0, 0.9
*FILM PROPERTY, NAME=filmp
 2.0,  80.0
 2.3,  90.0
 8.5, 180.0
 …
*STEP
** For an Abaqus/Standard analysis:
*HEAT TRANSFER
** For an Abaqus/Explicit analysis:
*DYNAMIC TEMPERATURE-DISPLACEMENT, EXPLICIT
 …
*FILM, SINK AMPLITUDE=sink
 3, F2, 90.0, filmp
```

A uniform, temperature-dependent film coefficient and a time-dependent sink temperature can be defined for node 2, where the nodal area is 50, by

```
*AMPLITUDE, NAME=sink
0.0, 0.5, 1.0, 0.9
*FILM PROPERTY, NAME=filmp
 2.0,  80.0
 2.3,  90.0
 8.5, 180.0
 …
*STEP
** For an Abaqus/Standard analysis:
*HEAT TRANSFER
** For an Abaqus/Explicit analysis:
*DYNAMIC TEMPERATURE-DISPLACEMENT, EXPLICIT
 …
*CFILM, SINK AMPLITUDE=sink,
 2, 50, 90.0, filmp
```

Defining Nonuniform Film Conditions in a User Subroutine

A nonuniform film coefficient can be defined as a function of position, time, temperature, etc. in user subroutine *FILM* in Abaqus/Standard and in user subroutine *VFILM* in Abaqus/Explicit for element-based, surface-based, as well as node-based film conditions. If a nonuniform film is prescribed, SINK AMPLITUDE and FILM AMPLITUDE references are used only to modify the sink temperature and film coefficient that are passed into the user subroutine.

Input File Usage: Use the following option to define a nonuniform film coefficient for an element-based film condition:

**FILM*
element number or element set name, F*n*NU

Use the following option to define a nonuniform film coefficient for a surface-based film condition:

**SFILM*
surface name, FNU

Use the following option to define a nonuniform film coefficient for a node-based film condition:

**CFILM*, USER
node number or node set name, *nodal area*

Abaqus/CAE Usage: Element-based film conditions to define a nonuniform film coefficient are not supported in Abaqus/CAE. However, similar functionality is available using surface-based film conditions. Use the following option to define a nonuniform film coefficient for a surface-based film condition:

Interaction module: **Create Interaction**: **Surface film condition**: select region: **Definition**: **User-defined**

Use the following option to define a nonuniform film coefficient for a node-based film condition:

Interaction module: **Create Interaction**: **Concentrated film condition**: select region: **Definition**: **User-defined**

Prescribing Boundary Radiation

Heat flux on a surface due to radiation to the environment is governed by

$$q = -\sigma\epsilon\left[\left(\theta - \theta^Z\right)^4 - \left(\theta^0 - \theta^Z\right)^4\right],$$

where

q

is the heat flux across the surface,

ϵ

is the emissivity of the surface,

σ

is the Stefan-Boltzmann constant,

θ

is the temperature at this point on the surface,

θ^0

is an ambient temperature value, and

θ^Z

is the value of absolute zero on the temperature scale being used.

Heat flux due to radiation can be defined on element faces, on surfaces, or at nodes.

In the presence of a user-defined radiation amplitude, A_{rad}, and ambient amplitude, A_{amb}, the heat flux due to radiation is given as:

$$q = -A_{rad}\sigma\epsilon\left[\left(\theta - \theta^Z\right)^4 - \left(A_{amb}\theta^0 - \theta^Z\right)^4\right],$$

and Abaqus computes the effective ambient temperature written to the output database as $A_{amb}\theta^0$.

In the absence of a user-defined radiation amplitude, Abaqus ramps down the heat flux due to radiation that is present at the start of the analysis (or at the end of the previous step in a multistep analysis) using the default step amplitude. This enables a smooth transition between steps and improves solver convergence. Here, the heat flux is:

$$q = -\sigma\epsilon\left[\left(\theta - \theta^Z\right)^4 - \left(A_{amb}\theta^0 - \theta^Z\right)^4\right] + \left(1 - A_{step}\right)q_0,$$

where

q_0

is the initial heat flux at the start of a step and computed as $q_0 = -\sigma\epsilon\left[\left(\theta_{ini} - \theta^Z\right)^4 - \left(\theta^0_{ini} - \theta^Z\right)^4\right]$,

A_{step}

is the default amplitude for a procedure that can either change linearly with time throughout the step (ramp function) or be applied immediately and remains constant throughout the step (step function)—see *Defining Time Varying Prescribed Conditions*,

$\theta^0{}_{ini}$

is the ambient temperature at the end of the previous step or at the start of the analysis, and

θ_{ini}

is the temperature at the end of the previous step or at the start of the analysis.

The effective ambient temperature written to the output database is

$$\theta^Z + \left\{\left[\left(A_{amb}\theta^0 - \theta^Z\right)^4 + \left(1 - A_{step}\right)\frac{q_0}{\sigma\epsilon}\right]\right\}^{1/4},$$

Specifying Element-Based Radiation

To specify element-based radiation within a heat transfer or coupled temperature-displacement step definition, you must provide the ambient temperature value, θ^0, and the emissivity of the surface, ϵ. The radiation is applied to element edges in two dimensions and to element faces in three dimensions. The edge or face of the element on which the radiation occurs is identified by a radiation type label depending on the element type (see *About the Element Library*).

Input File Usage: **RADIATE*

element number or element set name, R*n*, θ^0, ϵ

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface radiation**: select region: **Radiation type: To ambient**, **Emissivity distribution:** select an analytical field, **Emissivity:** ϵ, and **Ambient temperature:** θ^0

Specifying Element-Based Radiation Conditions on Evolving Faces of an Element in Abaqus/Standard

To specify element-based radiation on three-dimensional continuum elements that support the temperature degree of freedom, you must provide the ambient temperature value, θ^0, and the emissivity of the surface, ϵ. The radiation is applied to element faces in three dimensions. The face of the element on which the radiation is to be placed is automatically identified at the start of an increment. When elements are added or removed using model change during an analysis or using element activation or element deletion during an increment of a step, the radiation boundary condition is automatically applied at the start of an increment on the new exposed faces and removed from the nonexposed faces. You must specify the element number or element set name and the radiation load type label. (see *About the Element Library*).

By default, radiation is applied on the exposed full element facet area. When you use partial element activation (see *Progressive Element Activation*), you can use user subroutine *UEPACTIVATIONFACET* to modify the exposed area over which radiation is specified. When elements are partially activated, you can apply radiation on the activated facet areas C-I1-I4, C-B-I2-I1, and B-I3-I2 by specifying the area fraction per element facet. On the internal cut area I1-I2-I3-I4 of the element as shown in *Figure 2*, you can use user subroutine *UEPACTIVATIONFACET* to specify the exposed internal surface area. Radiation is applied on the prescribed internal cut surface area.

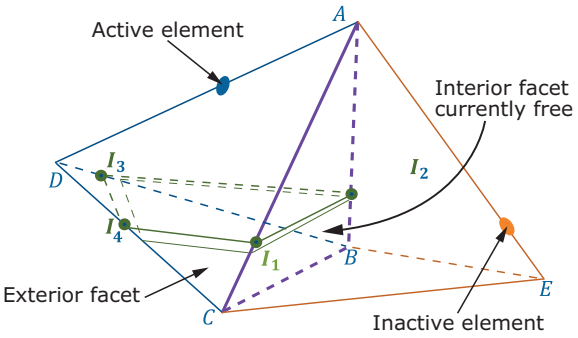


Figure 2: Partial facets and internal free surface for radiation.

Input File Usage: **RADIATE*

element number or element set name, RFS, θ^0, ϵ

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface radiation**: select region: **Radiation type: To ambient**, **Emissivity distribution:** select an analytical field, **Emissivity:** ϵ, and **Ambient temperature:** θ^0

Select **Toggle on dynamic surface activation**.

Specifying Surface-Based Radiation to Ambient

You can apply the radiation to a surface rather than to individual element faces. The surface that contains the element and face information is defined as described in *Element-Based Surface Definition*. You must specify the surface name; the radiation load type label, R; the ambient temperature value, θ^0; and the emissivity of the surface, ϵ.

By default, the specified emissivity is set to zero on portions of surfaces with active contact (closed status) to model conditions where the opposing surface is fully reflecting the radiating heat back to the surface. When the contact status changes with time (such as in a thermomechanical analysis), Abaqus uses the contact status distribution from the previous increment to determine portions of active contact. To accommodate a changing emissivity due to contact status changes and ensure robustness, Abaqus/Standard ramps the change to the emissivity over a time period that is a small fraction of the current step duration. The emissivity changes to zero the increment after contact becomes locally active and is reestablished the increment after contact becomes locally active in Abaqus/Explicit.

Optionally, you can turn off the dependence of emissivity on the contact status. If there is no contact in the model, Abaqus ignores the contact status dependence.

Input File Usage: Use the following option to define default surface-based radiation:

**SRADIATE*
surface name, R, θ^0, ϵ

Use the following option to explicitly define surface-based radiation to depend on the contact status (default):

**SRADIATE*, TRANSITION=TOUCHING
surface name, R, θ^0, ϵ, time period over which a change to emissivity is ramped

Use the following option to turn off contact status dependence for surface-based radiation:

**SRADIATE*, TRANSITION=NONE
surface name, R, θ^0, ϵ

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface radiation**: select region: **Radiation type: To ambient**, **Emissivity distribution: Uniform**, **Emissivity: ϵ**, and **Ambient temperature: θ^0**

Specifying Node-Based Radiation to Ambient

To specify node-based radiation within a heat transfer or coupled temperature-displacement step definition, you must provide the nodal area for a specified node number or node set; the ambient temperature value, θ^0; and the emissivity of the surface, ϵ. The associated degree of freedom is 11. For shell elements where the concentrated radiation is associated with a degree of freedom other than 11, you can specify the required data for a duplicate node that is constrained to the appropriate degree of freedom of the shell node by using an equation constraint.

Input File Usage: **CRADIATE*
node number or node set name, *nodal area*, θ^0, ϵ

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Concentrated radiation to ambient**: select region: **Associated nodal area: Emissivity:** ϵ and **Ambient temperature:** θ^0

Specifying Node-Based Radiation to Ambient at Phantom Nodes for Enriched Elements

For an enriched element (see *Modeling Discontinuities as an Enriched Feature Using the Extended Finite Element Method*), you can define the nodal area; the ambient temperature value, θ^0; and the emissivity of the surface, ϵ, at a phantom node that is originally located coincident with the specified real node.

Alternatively, you can define the nodal area; the ambient temperature value, θ^0; and the emissivity of the surface, ϵ, at a phantom node located at an element edge between two specified real corner nodes. This setting applies only to nodes with both pore pressure and temperature degrees of freedom.

Input File Usage: Use the following option to specify radiation conditions at a phantom node originally located coincident with the specified real node:

**CRADIATE*, PHANTOM=NODE
node number, *nodal area*, θ^0, ϵ

Use the following option to specify radiation conditions at a phantom located at an element edge:

**CRADIATE*, PHANTOM=EDGE
first corner node number, *second corner node number*, *nodal area*, θ^0, ϵ

Abaqus/CAE Usage: Specifying radiation conditions at phantom nodes for enriched elements is not supported in Abaqus/CAE.

Specifying Time-Dependent Radiation

The user-specified value of the ambient temperature, θ^0 and the radiation flux, can be varied throughout the step by referring to an amplitude definition. See *About Loads* and *Amplitude Curves* for details.

Input File Usage: Use the following options to define time-dependent film conditions:

**AMPLITUDE*, NAME=*temp_amp*
**AMPLITUDE*, NAME=*rad_amp*
**RADIATE*, AMBIENT AMPLITUDE=*temp_amp*, RADIATION AMPLITUDE=*rad_amp*
**SRADIATE*, AMBIENT AMPLITUDE=*temp_amp*, RADIATION AMPLITUDE=*rad_amp*
**CRADIATE*, AMBIENT AMPLITUDE=*temp_amp*, RADIATION AMPLITUDE=*rad_amp*

Specifying Average-Temperature Radiation Conditions

The average-temperature radiation condition is an approximation to the cavity radiation problem, where the radiative flux per unit area into a facet is

$$q_i^c = \sigma \epsilon_i \quad \left(\theta_{AVG}^4 - \left(\theta_i - \theta^Z\right)^4\right),$$

with the average temperature for the surface θ_{AVG} being calculated as

$$\theta_{AVG}^4 = \frac{1}{A_{total}} \sum_{j=1}^{N} A_j \left(\theta_j - \theta^Z\right)^4.$$

The average temperature in the cavity is computed at the beginning of each increment and held constant over the increment. Therefore, the average-temperature radiation condition has some dependency on the increment size, and you need to ensure that the increment size you use is appropriate for your model. If you see large changes in temperature over an increment, you might need to reduce the increment size. This option can only be used in three-dimensional analyses.

Input File Usage: Use the following option to define the average-temperature radiation condition on a surface:

**SRADIATE*
surface name, AVG, , ϵ

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface radiation**: select the surface region: **Radiation type: Cavity approximation (3D only)**, **Emissivity:** ϵ

Specifying the Value of Absolute Zero

You can specify the value of absolute zero, θ^Z, on the temperature scale being used; you must specify this value as model data. By default, the value of absolute zero is 0.0.

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 Value of the Stefan-Boltzmann Constant

If boundary radiation is prescribed, you must specify the Stefan-Boltzmann constant, σ; this value must be specified as model data.

Input File Usage: **PHYSICAL CONSTANTS*, STEFAN BOLTZMANN=σ

Abaqus/CAE Usage: Any module: **Model->Edit Attributes->*model_name***: **Stefan-Boltzmann constant:** σ

Modifying or Removing Boundary Radiation

Boundary radiation conditions can be added, modified, or removed as described in *About Loads*.

Electromagnetic Loads

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Prescribed Conditions*
- *About Loads*
- **CECHARGE*
- **CECURRENT*
- **DECHARGE*
- **DECURRENT*
- **DSECHARGE*
- **DSECURRENT*
- *Defining a concentrated current*
- *Defining a surface current*
- *Defining a body current*
- *Defining a surface current density*
- *Defining a body current density*
- *Defining a concentrated charge*
- *Defining a surface charge*
- *Defining a body charge*

Overview

You can apply electromagnetic loads in piezoelectric, coupled thermal-electrical, fully coupled thermal-electrical-structural, eddy current, and magnetostatic analyses.
For more information, see *Piezoelectric Analysis*, *Coupled Thermal-Electrical Analysis*, *Fully Coupled Thermal-Electrical-Structural Analysis*, *Eddy Current Analysis*, and *Magnetostatic Analysis*.

The types of electromagnetic loads available depend on the analysis being performed, as described in the sections below. See *About Loads* for general information that applies to all types of loading.

Defining Time-Dependent Electromagnetic Loads

The prescribed magnitude of a concentrated or a distributed electromagnetic load can vary with time during a step according to an amplitude definition, as described in *About Prescribed Conditions*. If different variations are needed for different loads, each load can refer to its own amplitude definition.

In a time-harmonic eddy current analysis all loads are assumed to be time-harmonic.

Modifying Electromagnetic Loads

Concentrated or distributed electromagnetic loads can be added, modified, or removed as described in *About Loads*.

Prescribing Electromagnetic Loads for Piezoelectric Analyses

In a piezoelectric analysis a concentrated electric charge can be prescribed at nodes, a distributed electric surface charge can be defined on element faces and surfaces, and a distributed electric body charge can be defined on elements.

Specifying Concentrated Electric Charge

To specify a concentrated electric charge, specify the node or node set and the magnitude of the charge.

Input File Usage: **CECHARGE*
node number or node set name, , charge magnitude

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Concentrated charge** for the **Types for Selected Step**; **Magnitude**: *charge magnitude*

Specifying Element-Based Distributed Electric Charge

You can specify a distributed surface charge (on element faces) or a distributed body charge (charge per unit volume). For an element-based surface charge you must identify the face of the element upon which the charge is prescribed in the charge label. The distributed charge types available depend on the element type. *About the Element Library* lists the distributed charges that are available for particular elements.

Input File Usage: **DECHARGE*
element number or element set name, charge label, charge magnitude

where *charge label* is ES*n* or EBF

Abaqus/CAE Usage: Use the following input to define a distributed surface charge on element faces:

Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Surface charge** for the **Types for Selected Step**; **Distribution**: select an analytical field, **Magnitude**: *charge magnitude*

Use the following input to define a body charge:

Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Body charge** for the **Types for Selected Step**

Specifying Surface-Based Distributed Electric Charge

When you specify a distributed electric charge on a surface, the element-based surface (see *Element-Based Surface Definition*) contains the element and face information. You must specify the surface name, the electric charge label, and the electric charge magnitude.

Input File Usage: **DSECHARGE*
surface name, ES, *charge magnitude*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Surface charge** for the **Types for Selected Step**; **Distribution**: **Uniform**, **Magnitude**: *charge magnitude*

Specifying Electric Charge in Direct-Solution Steady-State Dynamics Analysis

In the direct-solution steady-state dynamics procedure, electric charges are given in terms of their real and imaginary components.

Input File Usage: Use the following options to define electric charges in direct-integration steady-state dynamics analysis:

**CECHARGE*, REAL or IMAGINARY *(real or imaginary component)*
**DECHARGE*, REAL or IMAGINARY
**DSECHARGE*, REAL or IMAGINARY

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Concentrated charge**, **Surface charge**, or **Body charge** for the **Types for Selected Step**; **Magnitude**: *real component* + *imaginary component*

Loading in Mode-Based and Subspace-Based Procedures

Electrical charge loads should be used only in conjunction with residual modes in the eigenvalue extraction step, due to the "massless" mode effect. Since the electrical potential degrees of freedom do not have any associated mass, these degrees of freedom are essentially eliminated (similar to Guyan reduction or mass condensation) during the eigenvalue extraction. The residual modes represent the static response corresponding to the electrical charge loads, which will adequately represent the potential degree of freedom in the eigenspace.

Prescribing Electromagnetic Loads for Coupled Thermal-Electrical and Fully Coupled Thermal-Electrical-Structural Analyses

In a coupled thermal-electrical analysis and fully coupled thermal-electrical-structural analysis a concentrated current can be prescribed at nodes, distributed current densities can be defined on element faces and surfaces, and distributed body currents can be defined on elements.

Specifying Concentrated Current Density

To define concentrated currents, specify the node or node set and the magnitude of the current.

Input File Usage: **CECURRENT*
node number or node set name, , current magnitude

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Concentrated current** for the **Types for Selected Step**; **Magnitude**: *current magnitude*

Specifying Element-Based Distributed Current Density

You can specify distributed surface current densities (on element faces) or distributed body current densities (current per unit volume). For element-based surface current densities you must identify the face of the element

upon which the current is prescribed in the current label. The distributed current types available depend on the element type. *About the Element Library* lists the distributed current densities that are available for particular elements.

Input File Usage: **DECURRENT*
element number or element set name, current density label, current density magnitude

where the current density label is CS*n*, CS1, CS2, or CBF

Abaqus/CAE Usage: Use the following input to define a distributed surface current density on element faces:

Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Surface current** for the **Types for Selected Step**; **Distribution**: select an analytical field, **Magnitude**: *current density magnitude*

Use the following input to define a body current density:

Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Body current** for the **Types for Selected Step**

Specifying Surface-Based Distributed Current Densities

When you specify distributed current densities on a surface, the element-based surface (see *Element-Based Surface Definition*) contains the element and face information. You must specify the surface name, the current density label, and the current density magnitude.

Input File Usage: **DSECURRENT*
surface name, CS, *current density magnitude*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Surface current** for the **Types for Selected Step**: **Distribution**: **Uniform**, **Magnitude**: *current density magnitude*

Defining Nonuniform Current Density Magnitude in a User Subroutine

Nonuniform element volume, element facet, and nonuniform surface current density magnitude can be defined with user subroutine *DECURRENT*. If the magnitude is given, the value is passed into the user subroutine.

Input File Usage: Use the following option to define nonuniform element-based current density magnitude:

**DECURRENT*
element number or element set name, current density label,current density magnitude, table collection name

where the current density label is CS*n*NU (CS1NU, CS2NU, CS3NU, CS4NU, CS5NU, CS6NU),or CBFNU.

Use the following option to define nonuniform surface-based current density magnitude:

**DSECURRENT*
surface name, CSNU, *current density magnitude, table collection name*

Abaqus/CAE Usage: Use the following option to define nonuniform volume current density:

Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Body current density** for the **Types for Selected Step**; **Distribution**: **User-defined**

Use the following option to define nonuniform surface current density:

Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Surface current density** for the **Types for Selected Step**; **Distribution**: **User-defined**

Prescribing Electromagnetic Loads for Eddy Current and/or Magnetostatic Analyses

In an eddy current analysis a distributed surface current density vector can be defined on surfaces and a distributed volume current density vector can be defined on elements.

Specifying Element-Based Distributed Current Density Vectors

When you define a distributed volume current density vector, you must specify the element or element set, the current density vector label, the magnitude of the current density vector, the vector components of the current density, and an optional orientation name that defines the local coordinate system in which the vector components are specified. By default, the vector components of the current density are defined with respect to the global directions.

The specified current density vector direction components are normalized by Abaqus and, thus, do not contribute to the magnitude of the load.

Input File Usage: **DECURRENT*
element number or element set name, CJ, *current density vector magnitude, current density vector direction components, orientation name*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Body current density** for the **Types for Selected Step**; **Distribution**: **Uniform**

Specifying Surface-Based Distributed Current Density Vectors

When you specify distributed current density vectors on a surface, the element-based surface (see *Element-Based Surface Definition*) contains the element and face information. You must specify the surface name, the current density vector label, and the magnitude of the current density vector, the vector components of the current density, and an optional orientation name that defines the local coordinate system in which the surface current density is specified. By default, the vector components of the current density are defined with respect to the global directions.

The specified current density vector direction components are normalized by Abaqus and, thus, do not contribute to the magnitude of the load.

Input File Usage: **DSECURRENT*
surface name, CK, *current density vector magnitude, current density vector direction components, orientation name*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Surface current density** for the **Types for Selected Step**; **Distribution**: **Uniform**

Defining Nonuniform Current Density Vectors in a User Subroutine

Nonuniform volume current density vectors can be defined with user subroutine *UDECURRENT*, and nonuniform surface current density vectors can be defined with user subroutine *UDSECURRENT*. If the magnitude and direction components are given, the values are passed into the user subroutine.

Input File Usage: Use the following option to define nonuniform element-based current density vectors:

**DECURRENT*
element number or element set name, CJNU, *current density vector magnitude, current density vector direction components, orientation name*

Use the following option to define nonuniform surface-based current density vectors:

**DSECURRENT*
surface name, CKNU, *current density vector magnitude, current density vector direction components, orientation name*

Abaqus/CAE Usage: Use the following option to define nonuniform volume current density:

Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Body current density** for the **Types for Selected Step**; **Distribution**: **User-defined**

Use the following option to define nonuniform surface current density:

Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Surface current density** for the **Types for Selected Step**; **Distribution**: **User-defined**

Specifying Real and Imaginary Components of Current Density Vectors in a Time-Harmonic Eddy Current Analysis

In a time-harmonic eddy current analysis, current density vectors are given in terms of their real (in-phase) and imaginary (out-of-phase) components.

Input File Usage: Use the following options to define current density vectors:

**DECURRENT*, REAL or IMAGINARY
**DSECURRENT*, REAL or IMAGINARY

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical/Magnetic** for the **Category** and **Body current density** or **Surface current density** for the **Types for Selected Step**; *real components + imaginary components*

Acoustic and Shock Loads

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Loads*
- *Coupled Acoustic-Structural Analysis*
- **AMPLITUDE*
- **BOUNDARY*
- **CLOAD*
- **CONWEP CHARGE PROPERTY*
- **IMPEDANCE*
- **IMPEDANCE PROPERTY*
- **INCIDENT WAVE*
- **INCIDENT WAVE FLUID PROPERTY*
- **INCIDENT WAVE INTERACTION*
- **INCIDENT WAVE INTERACTION PROPERTY*
- **INCIDENT WAVE PROPERTY*
- **INCIDENT WAVE REFLECTION*
- **SIMPEDANCE*
- **UNDEX CHARGE PROPERTY*
- *Defining acoustic impedance*
- *Defining incident waves*
- *Defining an acoustic impedance interaction property*
- *Defining an incident wave interaction property*

Overview

You can apply acoustic loads in transient and steady-state dynamic analysis procedures to model boundary impedance, nonreflecting boundaries, concentrated pressure-conjugate loads, and incident wave loading.

The following types of acoustic loads are available:

- Boundary impedance defined on element faces or on surfaces.
- Nonreflecting radiation boundaries in exterior problems such as a structure vibrating in an acoustic medium of infinite extent.
- Concentrated pressure-conjugate loads prescribed at acoustic element nodes.
- Temporally and spatially varying pressure loading on acoustic and solid surfaces due to incident waves traveling through the acoustic medium.

Specified Boundary Impedance

A boundary impedance specifies the relationship between the pressure of an acoustic medium and the normal motion at the boundary. Such a condition is applied, for example, to include the effect of small-amplitude

"sloshing" in a gravity field or the effect of a compressible, possibly dissipative, lining (such as a carpet) between an acoustic medium and a fixed, rigid wall or structure.

The impedance boundary condition at any point along the acoustic medium surface is governed by

$$\dot{u}_{out} = \frac{1}{k_1}\dot{p} + \frac{1}{c_1}p,$$

where

$\dot{u}_{out}$

is the acoustic particle velocity in the outward normal direction of the acoustic medium surface,

p

is the acoustic pressure,

$\dot{p}$

is the time rate of change of the acoustic pressure,

$1/k_1$

is the proportionality coefficient between the pressure and the displacement normal to the surface, and

$1/c_1$

is the proportionality coefficient between the pressure and the velocity normal to the surface.

This model can be conceptualized as a spring and dashpot in series placed between the acoustic medium and a rigid wall. The spring and dashpot parameters are k_1 and c_1, respectively, defined per unit area of the interface surface. These reactive acoustic boundaries can have a significant effect on the pressure distribution in the acoustic medium, in particular if the coefficients k_1 and c_1 are chosen such that the boundary is energy absorbing. If no impedance, loads, or fluid-solid coupling are specified on the surface of an acoustic mesh, the acceleration of that surface is assumed to be zero. This is equivalent to the presence of a rigid wall at that boundary.

Use of the subspace-based steady-state dynamics procedure is not recommended if reactive acoustic boundaries with strong absorption characteristics are used. Since the effect of c_1 is not taken into account in an eigenfrequency extraction step, the eigenmodes might have shapes that are significantly different from the exact solution.

Sloshing of a Free Surface

To model small-amplitude "sloshing" of a free surface in a gravity field, set $1/k_1 = 1/(\rho_f g)$ and $1/c_1 = 0$, where ρ_f is the density of the fluid and g is the gravitational acceleration (assumed to be directed normal to the surface). This relation holds for small volumetric drag.

Acoustic-Structural Interface

The impedance boundary condition can also be placed at an acoustic-structural interface. In this case the boundary condition can be conceptualized as a spring and dashpot in series placed between the acoustic medium and the structure. The expression for the outward velocity still holds, with $\dot{u}_{out}$ now being the relative outward velocity of the acoustic medium and the structure:

$$\dot{u}_{out} = \mathbf{n} \cdot \left(\dot{\mathbf{u}}^f - \dot{\mathbf{u}}^m \right),$$

where $\dot{\mathbf{u}}^m$ is the velocity of the structure, $\dot{\mathbf{u}}^f$ is the velocity of the acoustic medium at the boundary, and $\mathbf{n}$ is the outward normal to the acoustic medium.

Steady-State Dynamics

In a steady-state dynamics analysis the expression for the outward velocity can be written in complex form as

$$\dot{u}_{out} = \left(\frac{1}{c_1} + i \frac{\Omega}{k_1} \right) p = \frac{1}{Z(\Omega)} p,$$

where Ω is the circular frequency (radians/second) and we define

$$\frac{1}{Z(\Omega)} \equiv \frac{1}{c_1} + i \frac{\Omega}{k_1}.$$

The term $1/Z(\Omega)$ is the complex admittance of the boundary, and $Z(\Omega)$ is its complex impedance. Thus, a required complex impedance or admittance value can be entered for a given frequency by specifying the parameters $1/c_1$ and $1/k_1$.

Specifying Impedance Conditions

You specify impedance coefficient data in an impedance property table. You can describe an impedance table in terms of the admittance parameters, $1/c_1$ and $1/k_1$, or in terms of the real and imaginary parts of the impedance. In the latter case Abaqus converts the user-defined table of impedance data to the admittance parameter form for the analysis.

The parameters in the table can be specified over a range of frequencies. The required values are interpolated from the table in steady-state harmonic response analysis only; for other analysis types, only the first table entry is used. The name of the impedance property table is referred to from a surface-based or element-based impedance definition. In Abaqus/CAE impedance conditions are always surface-based; surfaces can be defined as collections of geometric faces and edges or collections of element faces and edges.

In a steady-state dynamics analysis you cannot specify impedance conditions on a surface on which incident wave loading is applied.

Input File Usage: Use the following option to specify an impedance using a table of admittance parameters (default):

**IMPEDANCE PROPERTY*, NAME=*impedance property table name*, DATA=ADMITTANCE

Use the following option to specify an impedance using a table of the real and imaginary parts of the impedance:

**IMPEDANCE PROPERTY*, NAME=*impedance property table name*, DATA=IMPEDANCE

Abaqus/CAE Usage: Use the following input to specify an impedance using a table of admittance parameters:

Interaction module: **Create Interaction Property**: **Name**: *impedance property table name* and **Acoustic impedance**: **Data type**: **Admittance**

Use the following input to specify an impedance using a table of the real and imaginary parts of the impedance:

Interaction module: **Create Interaction Property**: **Name**: *impedance property table name* and **Acoustic impedance**: **Data type**: **Impedance**

Specifying Surface-Based Impedance Conditions

You can define the impedance condition on a surface. The impedance is applied to element edges in two dimensions and to element faces in three dimensions. The element-based surface (see *Element-Based Surface Definition*) contains the element and face information.

Input File Usage: **SIMPEDANCE*, PROPERTY=*impedance property table name*
surface name

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Acoustic impedance**: select surface: **Definition**: **Tabular**, **Acoustic impedance property**: *impedance property table name*

Specifying Element-Based Impedance Conditions

Alternatively, you can define the impedance condition on element faces. The impedance is applied to element edges in two dimensions and to element faces in three dimensions. The edge or face of the element on which the impedance is placed is identified by an impedance load type and depends on the element type (see *About the Element Library*).

Input File Usage: **IMPEDANCE*, PROPERTY=*impedance property table name*
element number or set name, *impedance load type label*

Abaqus/CAE Usage: Element-based impedance conditions are not supported in Abaqus/CAE. However, similar functionality is available using surface-based impedance conditions.

Modifying or Removing Impedance Conditions

Impedance conditions can be added, modified, or removed as described in *About Loads*.

Radiation Boundaries for Exterior Problems

An exterior problem such as a structure vibrating in an acoustic medium of infinite extent is often of interest. Such a problem can be modeled by using acoustic elements to model the region between the structure and a simple geometric surface (located away from the structure) and applying a radiating (nonreflecting) boundary condition at that surface. The radiating boundary conditions are approximate, so the error in an exterior acoustic analysis is controlled not only by the usual finite element discretization error but also by the error in the approximate radiation condition. In Abaqus the radiation boundary conditions converge to the exact condition in the limit as they become infinitely distant from the radiating structure. In practice, these radiation conditions provide accurate results when the surface is at least one-half wavelength away from the structure at the lowest frequency of interest.

Except in the case of a plane wave absorbing condition with zero volumetric drag, the impedance parameters in Abaqus/Standard are frequency dependent. The frequency-dependent parameters are used in the direct-solution and subspace-based steady-state dynamics procedures. In direct time integration procedures the zero-drag values

for the constants $1/c_1$ and $1/k_1$ are used. These values will give good results when the drag is small. (Small volumetric drag here means $\gamma \ll \rho_f \Omega$, where ρ_f is the density of the acoustic medium and Ω is the circular excitation frequency or sound wave frequency.)

A direct-solution steady-state dynamics procedure (*Direct-Solution Steady-State Dynamic Analysis*) must include both real and complex terms if nonreflecting (also called quiet) boundaries are present, because nonreflecting boundaries represent a form of damping in the system.

Several radiating boundary conditions are implemented as special cases of the impedance boundary condition. The details of the formulation are given in *Coupled acoustic-structural medium analysis*.

Element-based impedance conditions are not supported in Abaqus/CAE. However, similar functionality is available using surface-based impedance conditions.

Planar Nonreflecting Boundary Condition

The simplest nonreflecting boundary condition available in Abaqus assumes that the plane waves are normally incident on the exterior surface. This planar boundary condition ignores the curvature of the boundary and the possibility that waves in the simulation might impinge on the boundary at an arbitrary angle. The planar nonreflecting condition provides an approximation: acoustic waves are transmitted across such a boundary with little reflection of energy back into the acoustic medium. The amount of energy reflected is small if the boundary is far away from major acoustic disturbances and is reasonably orthogonal to the direction of dominant wave propagation. Thus, if an exterior (unbounded domain) problem is to be solved, the nonreflecting boundary should be placed far enough away from the sound source so that the assumption of normally impinging waves is sufficiently accurate. This condition would be used, for example, on the exhaust end of a muffler.

Input File Usage: Use either of the following options (default):

**SIMPEDANCE*, NONREFLECTING=PLANAR
**IMPEDANCE*, NONREFLECTING=PLANAR

Abaqus/CAE Usage: Use the following input to specify a surface-based planar nonreflecting boundary condition:

Interaction module: **Create Interaction**: **Acoustic impedance**: select surface: **Definition**: **Nonreflecting**, **Nonreflecting type**: **Planar**

Improved Nonreflecting Boundary Condition for Plane Waves

For the planar nonreflecting boundary condition to be accurate, the plane waves must be normally incident to a planar boundary. However, the angle of incidence is generally unknown in advance. A radiating boundary condition that is exact for plane waves with arbitrary angles of incidence is available in Abaqus. The radiating boundary can have any arbitrary shape. This boundary impedance is implemented only for transient dynamics.

Input File Usage: Use either of the following options:

**SIMPEDANCE*, NONREFLECTING=IMPROVED
**IMPEDANCE*, NONREFLECTING=IMPROVED

Abaqus/CAE Usage: Use the following input to specify a surface-based improved planar nonreflecting boundary condition:

Interaction module: **Create Interaction**: **Acoustic impedance**: select surface: **Definition**: **Nonreflecting**, **Nonreflecting type**: **Improved planar**

Geometry-Based Nonreflecting Boundary Conditions

Four other types of absorbing boundary conditions that take the geometry of the radiating boundary into account are implemented in Abaqus: circular, spherical, elliptical, and prolate spheroidal. These boundary conditions offer improved performance over the planar nonreflecting condition if the nonreflecting surface has a simple, convex shape and is close to the acoustic sources. The various types of absorbing boundaries are selected by defining the required geometric parameters for the element-based or surface-based impedance definition.

The geometric parameters affect the nonreflecting surface impedance. To specify a nonreflecting boundary that is circular in two dimensions or a right circular cylinder in three dimensions, you must specify the radius of the circle. To specify a nonreflecting spherical boundary condition, you must specify the radius of the sphere. To specify a nonreflecting boundary that is elliptical in two dimensions or a right elliptical cylinder in three dimensions or to specify a prolate spheroid boundary condition, you must specify the shape, location, and orientation of the radiating surface. The two parameters specifying the shape of the surface are the semimajor axis and the eccentricity. The semimajor axis, a, of an ellipse or prolate spheroid is analogous to the radius of a sphere: it is one-half the length of the longest line segment connecting two points on the surface. The semiminor axis, b, is one-half the length of the longest line segment that connects two points on the surface and is orthogonal to the semimajor axis line. The eccentricity, ϵ, is defined as $\epsilon = \sqrt{1 - (b/a)^2}$.

See *Acoustic radiation impedance of a sphere in breathing mode* and *Acoustic-structural interaction in an infinite acoustic medium* for benchmark problems showing the use of these conditions.

Input File Usage: Use one of the following options:

**SIMPEDANCE*, NONREFLECTING=CIRCULAR
**SIMPEDANCE*, NONREFLECTING=SPHERICAL
**SIMPEDANCE*, NONREFLECTING=ELLIPTICAL
**SIMPEDANCE*, NONREFLECTING=PROLATE SPHEROIDAL

In each case, the **IMPEDANCE* element-based option can be used instead of **SIMPEDANCE*.

Abaqus/CAE Usage: Use the following input to specify surface-based geometric nonreflecting boundary conditions:

Interaction module: **Create Interaction**: **Acoustic impedance**: select surface: **Definition**: **Nonreflecting**, **Nonreflecting type**: **Circular**, **Spherical**, **Elliptical**, or **Prolate spheroidal**

Combining Different Radiation Conditions in the Same Problem

Since the radiation boundary conditions for the different shapes are spatially local and do not involve discretization in the infinite exterior domain, an exterior boundary can consist of the combination of several shapes. The appropriate boundary condition can then be applied to each part of the boundary. For example, a circular cylinder can be terminated with hemispheres (see *Fully and sequentially coupled acoustic-structural analysis of a muffler*), or an elliptical cylinder can be terminated with prolate spheroidal halves. This modeling technique is most effective if the boundaries between surfaces are continuous in slope as well as displacement, although this is not essential.

Concentrated Pressure-Conjugate Load

Distributed “loads” on acoustic elements can be interpreted as normal pressure gradients per unit density (dimensions of force per unit mass or acceleration). When used in Abaqus, the applied distributed loads must be integrated over a surface area, yielding a quantity with dimensions of force times area per unit mass (or volumetric acceleration). For analyses in the frequency domain and for transient dynamic analyses where the volumetric drag is zero, this acoustic load is equal to the volumetric acceleration of the fluid on the boundary. For example, a horizontal, flat rigid plate oscillating vertically imposes an acceleration on the acoustic fluid and an acoustic “load” equal to this acceleration times the surface area of the plate. For the transient dynamics formulation in the presence of volumetric drag, however, the specified “load” is slightly different. It is also a force times area per unit mass; but this force effect is partially lost to the volumetric drag, so the resulting volumetric acceleration of the fluid on the boundary is reduced. Noting this distinction for the special case of volumetric drag and transient dynamics, it is nevertheless convenient to refer to acoustic “loads” as volumetric accelerations in general.

An inward volumetric acceleration can be applied by a positive concentrated load on degree of freedom 8 at a node of an acoustic element that is on the boundary of the acoustic medium. In Abaqus/Standard you can specify the in-phase (real) part of a load (default) and the out-of-phase (imaginary) part of a load. Inward particle accelerations (force per unit mass in transient dynamics) on the face of an acoustic element should be lumped to concentrated loads representing inward volumetric accelerations on the nodes of the face in the same way that pressure on a face is lumped to nodal forces on stress/displacement elements.

Input File Usage: Use the following option to define the real part of the load:

**CLOAD*, REAL

Use the following option to define the imaginary part of the load:

**CLOAD*, IMAGINARY

Abaqus/CAE Usage: Load module: **Create Load**: choose **Acoustic** for the **Category** and **Inward volume acceleration** for the **Types for Selected Step**

Incident Wave Loading due to External Sources

Abaqus provides a type of distributed load for loads due to external wave sources. Individual spherical monopole or individual or diffuse planar sources can be defined, subjecting the fluid and solid region of interest to an incident field of waves. Waves produced by an explosion or sound source propagate from the source, impinging on and passing over the structure, producing a temporally and spatially varying load on the structural surface. In the fluid the pressure field is affected by reflections and emissions from the structure as well as by the incident field from the source itself. The incident wave loads on acoustic and/or solid meshes depend on the location of the source node, the properties of the propagating fluid, and the reference time history or frequency dependence specified at the reference (“standoff”) node as indicated in *Figure 1*.

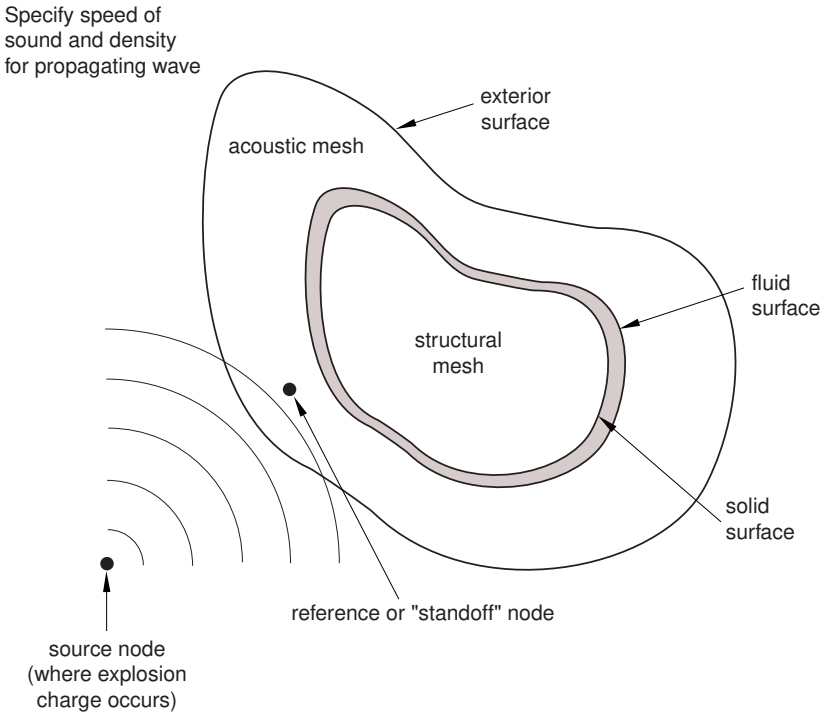


Figure 1: Incident wave loading model.

Several distinct modeling methods can be used in Abaqus with incident wave loading, requiring different approaches to applying the incident wave loads. For problems involving solid and structural elements only (for example, where the incident wave field is due to waves in air) the wave loading is applied roughly like a distributed surface load. This might apply to an analysis of blast loads in air on a vehicle or building (see *Example: Airblast Loading on a Structure*, shown in *Figure 6*). In Abaqus/Explicit the CONWEP model can be used for air blast loading on solid and structural elements, without the need to model the fluid medium. *Deformation of a sandwich plate under CONWEP blast loading* is an example of a blast loading problem.

Incident wave loads (with the exception of CONWEP loading) can be applied to beam structures as well; this is a common modeling method for ship whipping analysis and for steel frame buildings subject to blast loads. Incident wave loads can be applied to surfaces defined on two- or three-dimensional beam elements. However, incident wave loads can be applied only to three-dimensional beams for transient dynamic analysis where beam fluid inertia is defined. Incident wave loads cannot be defined on frame elements, line spring elements, three-dimensional open-section beam elements, or three-dimensional Euler-Bernoulli beams.

In underwater explosion analyses (for example, a ship or submerged vehicle subjected to an underwater explosion loading as depicted in *Figure 4* and *Figure 5*) the fluid is also discretized using a finite element model to capture the effects of the fluid stiffness and inertia. For these problems involving both solid and acoustic elements, two formulations of the acoustic pressure field exist. First, the acoustic elements can be used to model the total pressure in the medium, including the effects of the incident field and the overall system's response. Alternatively, the acoustic elements can be used to model only the response of the medium to the wave loads, not the wave pulse itself. The former case will be referred to as the “total wave” formulation, the latter as the “scattered wave” formulation.

Incident wave interactions are also used to model sound fields impinging on structures or acoustic domains. The acoustic field scattered by a structure or the sound transmitted through the structure might be of interest. Usually, sound scattering and transmission problems are modeled using the scattered formulation with steady-state dynamic procedures. Transient procedures can also be used, in a manner analogous to underwater explosion analysis problems.

Scattered and Total Wave Formulations

The distinction between the total wave formulation and the scattered wave formulation is relevant only when incident wave loads are applied. The total wave formulation is more closely analogous to structural loading than the scattered wave formulation: the boundary of the acoustic medium is specified as a loaded surface, and a time-varying load is applied there, which generates a response in the acoustic medium. This response is equal to the total acoustic pressure in the medium. The scattered wave formulation exploits the fact that when the acoustic medium is linear, the response in the medium can be decomposed into a sum of the incident wave and the scattered field. The total wave formulation must be used when the acoustic medium is nonlinear due to possible fluid cavitation (see *Loading due to an incident dilatational wave field*).

Table 1 describes the procedure types for which each formulation is supported.

Table 1: Supported procedures for scattered and total wave formulations.

Procedure	Scattered	Total Wave
Steady-state dynamics	Yes	No
Transient	Yes	Yes

Scattered Wave Formulation

When the mechanics of a fluid can be described as linear, the observed total acoustic pressure can be decomposed into two components: the known incident wave and the "scattered" wave that is produced by the interaction of the incident wave with structures and/or fluid boundaries. When this superposition is applicable, it is common practice to seek the "scattered" wave field solution directly. When using the scattered wave formulation, the pressures at the acoustic nodes are defined to be only the scattered part of the total pressure. Both acoustic and solid surfaces at the acoustic-structural interface should be loaded in this case.

When using incident wave loads in steady-state dynamic procedures, the scattered wave formulation must be used.

Input File Usage: Use the following option to specify the scattered wave formulation (default):

**ACOUSTIC WAVE FORMULATION*, TYPE=SCATTERED WAVE

Abaqus/CAE Usage: Any module: **Model->Edit Attributes->*model_name***. Toggle on **Specify acoustic wave formulation**: select **Scattered wave**

Total Wave Formulation

The total wave formulation (see *Coupled acoustic-structural medium analysis*) is particularly applicable when the acoustic medium is capable of cavitation, rendering the fluid mechanical behavior nonlinear. It should also be used if the problem contains either a curved or a finite extent boundary where the pressure history is prescribed. Only the outer acoustic surfaces should be loaded with the incident wave in this case, and the incident wave source must be located exterior to the fluid model. Any impedance or nonreflecting condition that might exist on this outer acoustic boundary applies only on the part of the acoustic solution that does not include the prescribed incident wave field (that is, only the scattered field is subject to the nonreflecting condition). Thus, the applied incident wave loading will travel into the problem domain without being affected by the nonreflecting conditions on the outer acoustic surface.

In the total wave formulation the acoustic pressure degree of freedom stands for the total dynamic acoustic pressure, including contributions from incident and scattered waves and, in Abaqus/Explicit, the dynamic effects of fluid cavitation. The pressure degree of freedom does not include the acoustic static pressure, which can be

specified as an initial condition (see *Defining Initial Acoustic Static Pressure*). This acoustic static pressure is used only in determining the cavitation status of the acoustic element nodes and does not apply any static loads to the acoustic or structural mesh at their common wetted interface. It does not apply to analyses using Abaqus/Standard.

Input File Usage: Use the following option to specify the total wave formulation:

**ACOUSTIC WAVE FORMULATION*, TYPE=TOTAL WAVE

Abaqus/CAE Usage: Any module: **Model->Edit Attributes->*model_name***. Toggle on **Specify acoustic wave formulation**: select **Total wave**

Initialization of Acoustic Fields

For transient dynamics, when the total wave formulation is used with the incident wave standoff point located inside the acoustic finite element domain, the acoustic solution is initialized to the values of the incoming incident wave. This initialization is performed automatically, for pressure-based incident wave amplitude definitions only, at the beginning of the first direct-integration dynamic step in an analysis; in restarted analyses, steps are counted from the beginning of the initial analysis. This initialization not only saves computational time but also applies the incident wave loading without significant numerical dissipation or distortion. During the initialization phase all incident wave loading definitions in the first dynamic analysis step are considered, and all acoustic element nodes are initialized to the incident wave field at time zero. Incident wave loads specified with different source locations count as separate load definitions for the purpose of initialization of the acoustic nodes. Any reflections of the incident wave loads are also taken into account during the initialization phase.

Describing Incident Wave Loading

To use incident wave loading, you must define the following:

- information that establishes the direction and other properties of the incident wave,
- the time history or frequency dependence of the source pulse at some reference (“standoff”) point,
- the fluid and/or solid surfaces to be loaded, and
- any reflection plane outside the problem domain, such as a seabed in an underwater explosion study, that would reflect the incident wave onto the problem domain.

Two interfaces are available in Abaqus for applying incident wave loads: a preferred interface that is supported in Abaqus/CAE and an alternative interface that has been available in previous releases and is not supported in Abaqus/CAE. The preferred interface is conceptually the same as the alternative interface and uses essentially the same data. The preferred interface options include the term “interaction” to distinguish them from the incident wave and incident wave property options of the alternative interface. Unless otherwise specified, the discussion in this section applies to both of the interfaces. The usages for the preferred interface are included in the discussion; the usages for the alternative interface are described in *Alternative Incident Wave Loading Interface* below. Refer to the example problems discussed at the end of this section to see how the incident wave loading is specified using the preferred interface.

Prescribing Geometric Properties and the Speed of the Incident Wave

You must refer to a property definition for each prescribed incident wave. Incident wave loads in Abaqus might be either planar, spherical, or diffuse. You select a planar incident wave (default), spherical incident wave, or a diffuse field in the incident wave property definition.

Planar incident waves maintain constant amplitude as they travel in space; consequently, the speed and direction of travel are the critical parameters to define. The speed is defined in the incident wave interaction property definition, and the direction is determined by the locations of the source and standoff points you define as part of the incident wave interaction.

For spherical incident wave definitions, the wave reduces in amplitude as a function of the distance from the source. By default, the amplitude of a spherical wave is inversely proportional to the distance from the source; this behavior is called "acoustic" propagation. For the preferred interface you can modify the default propagation behavior to define spatial decay of the incident wave field. The dimensionless constants A, B, and C are used to define the spatial decay as a function of the distance R_j between the source point and the loaded point and the distance R_0 between the source point and the standoff point:

$$p_x\left(R_0, R_j\right) \equiv \left(\frac{R_0}{R_j}\right)^{\left[\frac{(A+1)R_j}{CR_0+(B+1)R_j}\right]}.$$

Refer to *Loading due to an incident dilatational wave field* for details of the generalized spatial decay formulation.

In Abaqus incident wave interactions can be used to simulate diffuse incident fields. Diffuse fields are characteristic of reverberant spaces or other situations in which waves from many directions strike a surface. For example, reverberant chambers are constructed intentionally in acoustic test facilities for sound transmission loss measurements. The diffuse field model used in Abaqus, as shown in *Figure 2*, allows you to specify a seed number N; N^2 deterministic incident plane waves travel along vectors distributed over a hemisphere so that the incident power per solid angle approximates a diffuse incident field.

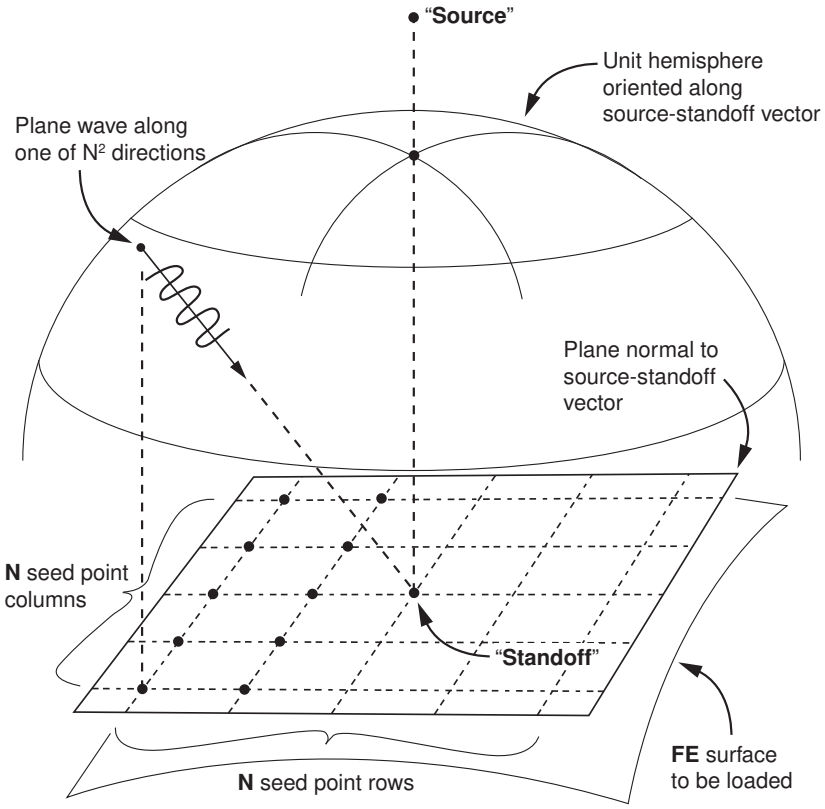


Figure 2: Diffuse loading model.

The fluid and the solid surfaces where the incident loading acts are specified in the incident wave loading definition. The incoming wave load is further described by the locations of its source point and of a reference ("standoff") point where the wave amplitude is specified. For information on how to specify these surfaces and

the standoff point, see *Identifying the Fluid and the Solid Surfaces for Incident Wave Loading* and *The Standoff Point* below. For a planar wave the specified locations of the source and the standoff points are used to define the direction of wave propagation.

The speed of the incident wave is prescribed by giving the properties for the incident wave-bearing acoustic medium. These specified properties should be consistent with the properties specified for the fluid discretized using acoustic elements.

For the preferred interface you must define nodes corresponding to the source and standoff points for the incident wave; the node numbers or set names must be specified for each incident wave definition. The node set names, if used, must contain only a single node. Neither the source node nor the standoff node should be connected to any elements in the model.

Input File Usage:

**INCIDENT WAVE INTERACTION PROPERTY*,
NAME=*wave property name*, TYPE=PLANE or SPHERE
speed of sound, fluid mass density, A, B, C
**INCIDENT WAVE INTERACTION*, PROPERTY=*wave property name*
fluid surface name, *source node*, *standoff node*, *reference magnitude*

The constants *A*, *B*, and *C* apply only for spherical incident waves with generalized spatial decay propagation.

**INCIDENT WAVE INTERACTION PROPERTY*,
NAME=*wave property name*, TYPE=DIFFUSE
speed of sound, fluid mass density, N
**INCIDENT WAVE INTERACTION*, PROPERTY=*wave property name*
fluid surface name, *source node*, *standoff node*, *reference magnitude*

The seed number *N* generates planar incident waves with directions distributed on a hemisphere centered at the standoff point.

Abaqus/CAE Usage:

Interaction module: **Create Interaction Property**: **Name**: *wave property name* and **Incident wave**, **Speed of sound in fluid**: *speed of sound*, **Fluid density**: *fluid mass density*

Select one of the following definitions:

Definition: **Planar**
Definition: **Spherical**, **Propagation model**: **Acoustic**
Definition: **Spherical**, **Propagation model**: **Generalized decay**,
enter values for **A**, **B**, and **C**
Definition: **Diffuse**, **Seed number**: *N*

Create Interaction: **Incident wave**: select the source point, select the standoff point, select the region: **Wave property**: *wave property name*, **Reference magnitude**: *reference magnitude*

Identifying the Fluid and the Solid Surfaces for Incident Wave Loading

In the scattered wave formulation the incident wave loading must be specified on all fluid and solid surfaces that reflect the incident wave with two exceptions:

- those fluid surfaces that have the pressure values directly prescribed using boundary conditions; and

- those fluid surfaces that have symmetry conditions (the symmetry must hold for both the loading and the geometry).

In problems with a fluid-solid interface both surfaces must be specified in the incident wave loading definition for the scattered formulation. See *Example: Submarine Close to the Free Surface*, shown in *Figure 4*.

When the total pressure-based formulation is specified, the incident wave loading must be specified only on the fluid surfaces that border the infinite region that is excluded from the model. Typically, these surfaces have a nonreflecting radiation condition specified on them, and the implementation ensures that the radiation condition is enforced only on the scattered response of the modeled domain and not on the incident wave itself. See *Example: Submarine Close to the Free Surface* and *Example: Surface Ship*, shown in *Figure 4* and *Figure 5*, respectively.

In certain problems, such as blast loads in air, you might decide that the blast wave loads on a structure need to be modeled, but the surrounding fluid medium itself does not. In these problems the incident wave loading is specified only on the solid surfaces since the fluid medium is not modeled. The distinction between the scattered wave formulation and the total wave formulation for handling the incident wave loading is not relevant in these problems since the wave propagation in the fluid medium is of no interest.

The Standoff Point

In transient analyses the standoff point is a reference point used to specify the pulse loading time history: it is the point at which the user-defined pulse history is assumed to apply with no time delay, phase shift, or spreading loss. In steady-state analyses using discrete planar or spherical sources, the standoff point is the point at which the incident field has zero phase.

In transient analyses the standoff point should be defined so that it is closer to the source than any point on the surfaces in the model that would reflect the incident wave. Doing so ensures that all the points on these surfaces will be loaded with the specified time history of the source and that the analysis begins before the wave overtakes any portion of these surfaces. To save analysis time, the standoff point is typically on or near the solid surface where the incoming incident wave would be first deflected (see *Example: Submarine Close to the Free Surface*, shown in *Figure 4*). However, the standoff point is a fixed point in the analysis: if the loaded surfaces move before the incident wave loading begins, due to previous analysis steps or geometric adjustments, the surfaces might envelop the specified standoff point. Care should be taken to define a standoff point such that it remains closer to the incident wave source point than any point on the loaded surfaces at the onset of the loading.

When the total wave formulation is used and the incident wave loading is specified in the first step of the analysis in terms of pressure history, Abaqus automatically initializes the pressure and the pressure rate at the acoustic nodes to values based on the incident wave loading. This allows the acoustic analysis to start with the incident waves partially propagated into the problem domain at time zero and assumes that this propagation had taken place with negligible effect of any volumetric dissipative sources such as the fluid drag. When the incident wave loading is specified in terms of the pressure values, the recommendations given above for selecting a standoff point are valid with the total wave formulation as well. However, when the incident wave loading is specified in terms of acceleration values, the automatic initialization is not done and the standoff point should be located near the exterior fluid boundary of the model such that the standoff point is closer to the source than any point on the exterior boundary. See *Example: Submarine Close to the Free Surface* and *Example: Surface Ship*, shown in *Figure 4* and *Figure 5*, respectively.

In steady-state analyses the role of the standoff point is somewhat different. When the incident wave interaction property is of planar or spherical type, you define the real and imaginary parts of the magnitude at the standoff point. Separately, the specified real and imaginary incident waves are taken to have zero phase at the standoff point (combined, these two waves could be equivalent to a single wave with nonzero phase at the standoff). Every location on the loaded surface has a phase shift in the applied pressure or acoustic traction, corresponding to the difference in propagation time between the loaded point and the standoff. This means that an incident wave defined, for example, with a pure real value at the standoff point generates both real and imaginary tractions at all the other points on the loaded surface.

When the incident wave is of diffuse type, the role of the standoff and source points is primarily to orient the loaded surface with respect to the incoming reverberant field. The model used for diffuse incident wave loading applies a set of deterministically defined plane waves, whose directions are defined as vectors connecting the standoff point and an array of points on a hemisphere. This hemisphere is centered at the standoff point, and its apex is the source point. The array of points is set according to the specified seed, N, and a deterministic algorithm that arranges N^2 points on the hemisphere. The algorithm concentrates the points so that the incident waves in the diffuse field model are concentrated at normal incidence, with fewer waves at oblique angles. The specified amplitude value and reference magnitude are divided equally among the N^2 incident waves. The orientation of the hemisphere containing the incident waves in the diffuse model is the same for all the points on the loaded surface—it does not vary with the local normal vector on the surface.

Defining the Amplitude of the Source Pulse

For transient analyses the time history to be specified by the user is that observed at the standoff point: histories at a point on the loaded surface are computed from the wave type and the location of that point relative to the standoff point. The time history of the acoustic source pulse can be defined either in terms of the fluid pressure values or the fluid particle acceleration values. Pressure time histories can be used for any type of element, such as acoustic, structural, or solid elements; acceleration time histories are applicable only for acoustic elements. In either case a reference magnitude is specified for any given incident-wave-loaded surface, and a reference to a time-history data table defined by an amplitude curve is specified. The reference magnitude varies with time according to the amplitude definition.

For steady-state dynamic analyses the amplitude definition specified as part of the incident wave interaction definition is interpreted as the frequency dependence of the wave at the standoff point.

Currently the source pulse description in terms of fluid particle acceleration history is limited to planar incident waves acting on fluid surfaces in transient analyses. Further, if an impedance condition is specified on the same fluid surface along with incident wave loading, the source pulse is restricted to the pressure history type even for planar incident waves. The source pulse in terms of pressure history can be used without these limitations; that is, pressure-history-based incident wave loading can be used with fluid or solid surfaces, with or without impedance, and for both planar and spherical incident waves.

When the source pulse is specified using pressure values and is applied on a fluid surface, the pressure gradient is computed and applied as a pressure-conjugate load on these surfaces. Hence, it is desirable to define the pulse amplitude to begin with a zero value, particularly when the cavitation in the fluid is a concern. If the structural response is of primary concern and the scattered formulation is being used, any initial jump in the pressure amplitude can be addressed by applying additional concentrated loads on the structural nodes that are tied to the acoustic mesh, corresponding to the initial jump in the incident wave pressure amplitude. Clearly, the additional load on any given structural node should be active from the instance the incident wave first arrives at that structural node. However, the scattered wave solution in the fluid still needs careful interpretation taking the initial jump into account.

Input File Usage: Use the following option to define the time history in terms of fluid pressure values:

**INCIDENT WAVE INTERACTION*, PRESSURE AMPLITUDE=*amplitude data table name*
solid or fluid surface name, *source node*, *standoff node*, *reference magnitude*

Use the following option to define the time history in terms of fluid particle acceleration values:

**INCIDENT WAVE INTERACTION*, ACCELERATION AMPLITUDE=*amplitude data table name*
fluid surface name, *source node*, *standoff node*, *reference magnitude*

Use the following option to define the real part of the loading (default):

**INCIDENT WAVE INTERACTION*, REAL

Use the following option to define the imaginary part of the loading:

**INCIDENT WAVE INTERACTION*, IMAGINARY

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Incident wave**: select the source point, select the standoff point, select the region: **Reference magnitude**: *reference magnitude*

Use the following options to define the time history in terms of fluid pressure values or fluid particle acceleration values:

Definition: **Pressure** or **Acceleration**, **Pressure amplitude** or **Acceleration amplitude**: *amplitude data table name*

Use the following options to define the real or imaginary part of the loading:

Toggle on **Real amplitude** and/or **Imaginary amplitude**: *amplitude data table name*

Defining Bubble Loading for Spherical Incident Wave Loading

An underwater explosion forms a highly compressed gas bubble that interacts with the surrounding water, generating an outward-propagating shock wave. The gas bubble floats upward as it generates these waves changing the relative positions of the source and the loaded surfaces. The loading effects due to bubble formation can be defined for spherical incident wave loading by using a bubble definition in conjunction with the incident wave loading definition.

The bubble dynamics can be described using a model internal to Abaqus or by using tabulated data. Abaqus has a built-in mechanical model of the bubble interacting with the surrounding fluid, which is simulated numerically to generate a set of data prior to running the finite element analysis. You can specify the explosive material parameters, ending time, and other parameters that affect the computation of the bubble amplitude curve used, as shown in *Table 2*.

Table 2: Parameters that define the bubble behavior.

Name	Dimensions	Description	Default
K	$FL^{-2}(LM^{-1/3})^{1+A}$	Charge constant	None
k	$T/(M^{\frac{1-B}{3}}L^{B})$	Charge constant	None
A	Dimensionless	Similitude spatial exponent	None
B	Dimensionless	Similitude temporal exponent	None
K_c	F/L^2	Charge constant	None
γ	Dimensionless	Ratio of specific heats for explosion gas	None
ρ_c	M/L^3	Charge material density	None

Name	Dimensions	Description	Default
m_c	M	Mass of charge	None
d_I	L	Initial charge depth	None
$\mathbf{n}_X$	Dimensionless	*X*-direction cosine of the free surface normal	None
$\mathbf{n}_Y$	Dimensionless	*Y*-direction cosine of the free surface normal	None
$\mathbf{n}_Z$	Dimensionless	*Z*-direction cosine of the free surface normal	None
g	L/T^2	Acceleration due to gravity	None
p_{atm}	F/L^2	Atmospheric pressure at free surface	None
η	Dimensionless	Wave effect parameter	1.0
C_D	Dimensionless	Bubble drag coefficient	0.0
E_D	Dimensionless	Bubble drag exponent	2.0
T_{final}	T	Maximum allowable time in bubble simulation	None
N_{steps}	Dimensionless	Maximum allowable number of steps in bubble simulation	1500
Ω_{rel}	Dimensionless	Relative error tolerance parameter for bubble simulation	1×10^{-11}
X_{abs}	Dimensionless	Absolute error tolerance parameter for bubble simulation	1×10^{-11}
β	Dimensionless	Error control exponent for bubble simulation	0.2
ρ_f	M/L^3	Fluid mass density	None
c_f	L/T	Fluid speed of sound	None

All of the parameters specified affect only the bubble amplitude; other physical parameters in the problem are independent. You can suppress the effects of wave loss in the bubble dynamics and introduce empirical flow drag, if desired. Detailed information about the bubble mechanical model is given in *Loading due to an incident dilatational wave field*.

In an underwater explosion event a bubble migrates upward toward, and possibly reaches, the free water surface. If the bubble migration reaches the free water surface during the specified analysis time, Abaqus applies loads of zero magnitude after this point.

Model data about the bubble simulation are written to the data (`.dat`) file. During an Abaqus/Standard analysis history data are written each increment to the output database (`.odb`) file. The history data include the radius of the bubble and the bubble depth below the free water surface. For reference, the pressure and acoustic load quantities at the standoff point are also written to the data file; these load terms include the direct plane-wave term and the spherical spreading (“afterflow”) effect (see *Loading due to an incident dilatational wave field*).

For the preferred interface the loading effects due to bubble formation can be defined for spherical incident wave loading using the UNDEX charge property definition. Because the bubble simulation uses spherical symmetry, the incident wave interaction property must define a spherical wave.

Input File Usage: Use the following options to specify loading effects due to bubble formation using the UNDEX charge property definition:

**INCIDENT WAVE INTERACTION PROPERTY*,
NAME=*wave property name*, TYPE=SPHERE
**UNDEX CHARGE PROPERTY*
data defining the UNDEX charge
**INCIDENT WAVE INTERACTION*, PROPERTY=*wave property name*,
UNDEX
fluid surface name, *source node*, *standoff node*, *reference magnitude*

Abaqus/CAE Usage: Use the following input to specify loading effects due to bubble formation using the UNDEX charge property definition:

Interaction module: **Create Interaction Property**: **Name**: *wave property name* and **Incident wave**: **Definition**: **Spherical**, **Propagation model**: **UNDEX charge**, enter data defining the UNDEX charge
Create Interaction: **Incident wave**: **Definition**: **UNDEX**, **Wave property**: *wave property name*, enter data defining the UNDEX charge

Use the following input to specify pressure at the standoff point using tabulated data:

Load or Interaction module: **Create Amplitude**: **Name**: *pressure*
and select **Tabular**
Interaction module: **Create Interaction**: **Incident wave**: select the standoff point: **Definition**: **Pressure**, **Pressure amplitude**: *pressure*

Use the following input to specify source node location time histories using tabulated data:

Load or Interaction module: **Create Amplitude**: **Name**: *name*
and select **Tabular**
Load module: **Create Boundary Condition**: select step: **Displacement/Rotation** or **Velocity/Angular velocity**: select the source node as the region and toggle on the degree or degrees of freedom, **Amplitude**: *name*

Modeling Incident Wave Loading on a Moving Structure

To model the effect of relative motion between a structure (such as a ship) and the wave source during the analysis using the preferred interface, the source node can be assigned a velocity. It is assumed that the entire fluid-solid model is moving at a velocity with respect to the source node during the loading and that the speed of the model's motion is low compared to the speed of propagation of the incident wave. That is, the effect of the speed of the source is neglected in the computation of the loads, but the change in position of the source is included. This is equivalent to assuming that the relative motion between the source and the model is at a low Mach number. Relative motion can be specified only for transient analyses.

In addition to prescribing boundary conditions at the source node, a small mass element must be defined at the source node.

Input File Usage: Use the following option to assign a velocity to the source node:

**BOUNDARY*, TYPE=DISPLACEMENT *or* VELOCITY, AMPLITUDE=*name*
source node, *degrees of freedom*

Abaqus/CAE Usage: Load module: **Create Boundary Condition**: select step: **Velocity/Angular velocity** or **Displacement/Rotation**: select regions and toggle on the degree or degrees of freedom, **Amplitude**: *name*

Specifying the Reflection Effects

The waves emanating from the source might reflect off plane surfaces, such as seabeds or sea surfaces, before reaching the specified standoff point. Thus, the incident wave loading consists of the waves arriving from a direct path from the source, as well as those arriving from reflections off the planes. In Abaqus an arbitrary number of these planes can be defined, each with its own location, orientation, and reflection coefficient.

If no reflection coefficient is specified, the plane is assumed to be nonreflective; a zero reflected pressure is applied. If a reflection coefficient is specified, the magnitude of the reflected waves are modified by the reflection coefficient Q according to the formula:

$$p_{reflected} = Qp_{incident}.$$

Only real values for Q are used.

The reflection planes are allowed only for incident waves that are defined in terms of fluid pressure values. Only one reflection off each plane is considered. If the effect of many successive reflections is important, these surfaces should be part of the finite element model. Reflection planes should not be used at a boundary of the finite element model if the total wave formulation is used, since in that case the incident wave will be reflected automatically by that boundary.

Input File Usage: Use the following option in conjunction with the **INCIDENT WAVE INTERACTION* option to define an incident wave reflection plane:

**INCIDENT WAVE REFLECTION*

Abaqus/CAE Usage: Incident wave reflections are not supported in Abaqus/CAE.

Boundary with Prescribed Pressure

The acoustic pressure degree of freedom at nodes of acoustic elements can be prescribed using a boundary condition. However, since you can use the nodal acoustic pressure in an Abaqus analysis to refer to the total pressure at that point or to only the scattered component, care must be exercised in some circumstances.

When the total wave formulation is used, a boundary condition alone is sufficient to specify a prescribed total dynamic pressure on a boundary.

In an analysis without incident wave loading, the nodal degree of freedom is generally equal to the total acoustic pressure at that point. Therefore, its value can be prescribed using a boundary condition in a manner consistent with other boundary conditions in Abaqus. For example, you can set the acoustic pressure at all of the nodes at a duct inlet to a prescribed amplitude to analyze the propagation of waves along the duct. The free surface of a body of water can be modeled by setting the acoustic pressure to zero at the surface.

When incident wave loading is used, the scattered wave formulation defines the nodal acoustic degree of freedom to be equal to the scattered pressure. Consequently, a boundary condition definition for this degree of freedom affects the scattered pressure only. The total acoustic pressure at a node is not directly accessible in this formulation. Specification of the total pressure in a scattered formulation analysis is nevertheless required in some instances (for example, when modeling a free surface of a body of water). In this case, one of the following methods should be used.

If the fluid surface with prescribed total pressure is planar, unbroken, and of infinite extent, an incident wave reflection plane and a boundary condition can be used together to model the fact that the total pressure is zero on the free surface. A "soft" incident wave reflection plane coincident with the free surface will make sure that the structure is subjected to the incident wave load reflected off the free surface. A boundary condition setting the acoustic pressure in the surface equal to zero will make sure that any scattered waves emitted by the structure are reflected properly. The scattered wave solution in the fluid must be interpreted taking into consideration the fact that the incident field now includes a reflection of the source as well. If the fluid surface with prescribed total pressure is planar but broken by an object, such as a floating ship, this modeling technique might still be applied. However, the reflected loads due to the incident wave are computed as if the reflection plane passes through the hull of the ship; this approximation neglects some diffraction effects and might or might not be applicable in all situations of interest.

Alternatively, the free surface condition of the fluid can be eliminated by modeling the top layer of the fluid using structural elements, such as membrane elements, instead of acoustic elements. The "structural fluid" surface and the "acoustic fluid" surface are then coupled using either a surface-based mesh tie constraint (*Mesh Tie Constraints*) or, in Abaqus/Standard, acoustic-structural interface elements; and the incident wave loading must be applied on both the "structural fluid" and the "acoustic fluid" surfaces. The material properties of the "structural fluid" elements should be similar to those of the adjacent acoustic fluid. In Abaqus/Explicit the thickness of the "structural fluid" elements must be such that the masses at nodes on either side of the coupling constraint are nearly equal. This modeling technique allows the geometry of the surface on which total pressure is to be prescribed to depart from an unbroken, infinite plane. As a secondary benefit of this technique, you can obtain the velocity profile on the free surface since the displacement degrees of freedom are now activated at the "structural fluid" nodes. If a nonzero pressure boundary condition is desired, it can be applied as a distributed loading on the other side of the "structural fluid" elements.

Input File Usage: Use the following options for the first modeling technique with the default scattered wave formulation:

**BOUNDARY*
**INCIDENT WAVE REFLECTION*

Use the following option for the second modeling technique with the default scattered wave formulation:

**TIE*
**INCIDENT WAVE INTERACTION*

Use the following option with the total wave formulation:

**BOUNDARY*

Abaqus/CAE Usage: Load module: **Create BC**: choose **Other** for the **Category** and **Acoustic pressure** for the **Types for Selected Step**

Defining Air Blast Loading for Incident Shock Waves Using the CONWEP Model in Abaqus/Explicit

An explosion in air forms a highly compressed gas mass that interacts with the surrounding air, generating an outward-propagating shock wave. The loading effects due to an explosion in air can be defined, for spherical incident waves (air blast) or hemispherical incident waves (surface blast), by empirical data provided by the CONWEP model in conjunction with the incident wave loading definition.

Unlike an acoustic wave, a blast wave corresponds to a shock wave with discontinuities in pressure, density, etc. across the wave front. *Figure 3* shows a typical pressure history of a blast wave.

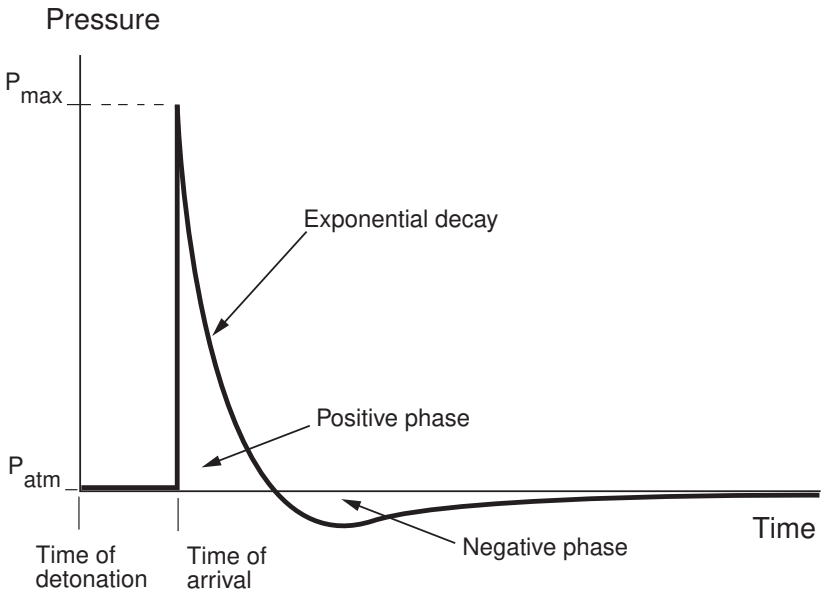


Figure 3: Pressure history of a blast wave.

The CONWEP model uses a scaled distance based on the distance of the loading surface from the source of the explosion and the amount of explosive detonated. For a given scaled distance, the model provides the following empirical data: the maximum overpressure (above atmospheric), the arrival time, the positive phase duration, and the exponential decay coefficient for both the incident pressure and the reflected pressure. Using these parameters, the entire time history of both the incident pressure and reflected pressure as shown in *Figure 3* can be constructed. Use of a standoff point is not required.

The total pressure, $P(t)$, on a surface due to the blast wave is a function of the incident pressure, $P_{incident}(t)$, the reflected pressure, $P_{reflect}(t)$, and the angle of incidence, θ, which is defined as the angle between the normal of the loading surface and the vector that points from the surface to the explosion source. The total pressure is defined as

$$P(t) = P_{incident}(t)\left[1 + \cos\theta - 2\cos^2\theta\right] + P_{reflect}(t)\cos^2\theta \quad for\ \cos\theta \geq 0;$$

$$P(t) = P_{incident}(t) \quad for\ \cos\theta < 0.$$

The air blast loading due to the total pressure can be scaled using a magnitude scale factor.

A detonation time can be specified if the explosion does not occur at the start of the analysis. The detonation time needs to be given in total time; see *Conventions* for a description of the time convention. The arrival time at a location is defined as the elapsed time for the wave to arrive at that location after detonation.

The CONWEP empirical data are given in a specific set of units, which must be converted to the units used in the analysis. You will need to specify multiplying factors for conversion of these units to SI units. For the specification of the mass of the explosive in TNT equivalence, you can choose any convenient mass unit, which can be different from the mass unit used in the analysis. For computation of the pressure loading, you will need to specify multiplying factors for conversion of length, time, and pressure units used in the analysis to SI units. Some typical conversion multiplier values are given in *Table 3*.

Table 3: Multipliers used in conjunction with the CONWEP model for conversion to SI units.

Quantity	Unit	SI Unit	Multiplier for conversion to SI
Mass	ton	kg	1000

Quantity	Unit	SI Unit	Multiplier for conversion to SI
Mass	lb	kg	0.45359
Length	mm	m	0.001
Length	ft	m	0.3048
Time	msec	sec	0.001
Pressure	MPa	Pa	10^6
Pressure	psi	Pa	6894.8
Pressure	psf	Pa	47.88

For any given amount of explosive, the CONWEP empirical data are valid only within a range of distances from the source. The minimum distance at which the data are valid corresponds to the charge radius. Thus, the analysis terminates if the distance of any part of the loading surface from the source is less than the charge radius. For distances that are larger than the maximum valid range, linear extrapolation is used up to an extended maximum range where the reflected pressure decreases to zero. No loading is applied beyond the extended maximum range.

The CONWEP empirical data do not account for shadowing by intervening objects or for any effects due to confinement. In the definition of incident wave interaction using the CONWEP model, you cannot use incident wave reflection.

The CONWEP pressure load can be requested as element face variable output to the output database file (see *Abaqus/Explicit Output Variable Identifiers*).

Input File Usage: Use the following options to specify loading effects due to explosion in air using the CONWEP charge property definition:

**INCIDENT WAVE INTERACTION PROPERTY*,
NAME=*wave property name*, TYPE=AIR BLAST or SURFACE BLAST
**CONWEP CHARGE PROPERTY*
data defining the CONWEP charge
**INCIDENT WAVE INTERACTION*, PROPERTY=*wave property name*,
CONWEP
loading surface name, *source node*, *detonation time*, *magnitude scale factor*

Abaqus/CAE Usage: Use the following options to specify loading effects due to explosion in air using the CONWEP charge property definition:

Interaction module: **Create Interaction Property**: **Name**: *wave property name* and **Incident wave**: **Definition**: **Air blast** or **Surface blast**: enter data defining the CONWEP charge

Interaction module: **Create Interaction**: **Name**: *incident wave name* and **Incident wave**: select the source point: **CONWEP (Air/Surface blast)**: select the region: **CONWEP Data**: enter data defining the time of detonation and magnitude scale factor

Modifying or Removing Incident Wave Loads

Only the incident wave loads that are specified in a particular step are applied in that step; previous definitions are removed automatically. Consequently, incident wave loads that are active during two subsequent steps should

be specified in each step. This is akin to the behavior that can be specified for other types of loads by releasing any load of that type in a step (see *About Loads*).

Alternative Incident Wave Loading Interface

In general, the concepts of the alternative incident wave loading interface are the same as the preferred interface; however, the syntax for specifying the incident wave loading is different. The preferred incident wave loading interface is supported in Abaqus/CAE. The alternative interface is not supported in Abaqus/CAE. For conceptual information, see *Incident Wave Loading due to External Sources*.

Prescribing the Geometric Properties and the Speed of the Incident Wave (Alternative Interface)

Conceptually, the alternative interface is the same as the preferred interface; however, the usages are different. For conceptual information, see *Prescribing Geometric Properties and the Speed of the Incident Wave*.

Input File Usage:

**INCIDENT WAVE PROPERTY*, NAME=*wave property name*,
TYPE=PLANE *or* SPHERE
data lines to specify the location of the acoustic source and the standoff point
**INCIDENT WAVE FLUID PROPERTY*
bulk modulus, *mass density*
**INCIDENT WAVE*, PROPERTY=*wave property name*

Abaqus/CAE Usage: The alternative incident wave loading interface is not supported in Abaqus/CAE.

Defining the Time History of the Source Pulse (Alternative Interface)

Conceptually, the alternative interface is the same as the preferred interface; however, the usages are different. For conceptual information, see *Defining the Amplitude of the Source Pulse*.

Input File Usage: Use the following option to define the time history in terms of fluid pressure values:

**INCIDENT WAVE*, PRESSURE AMPLITUDE=*amplitude data table name*
solid or fluid surface name, *reference magnitude*

Use the following option to define the time history in terms of fluid particle acceleration values:

**INCIDENT WAVE*, ACCELERATION AMPLITUDE=*amplitude data table name*
fluid surface name, *reference magnitude*

Abaqus/CAE Usage: The alternative incident wave loading interface is not supported in Abaqus/CAE.

Defining Bubble Loading for Spherical Incident Wave Loading (Alternative Interface)

Conceptually, the alternative interface is the same as the preferred interface; however, the usages are different. For conceptual information, see *Defining Bubble Loading for Spherical Incident Wave Loading*.

To define the bubble dynamics using a model internal to Abaqus, you can specify a bubble amplitude. Use of the bubble loading amplitude is generally similar to the use of any other amplitude in Abaqus.

Input File Usage: Use the following options:

*AMPLITUDE, DEFINITION=BUBBLE, NAME=*name*
*INCIDENT WAVE PROPERTY, TYPE=SPHERE,
NAME=*wave property name*
*INCIDENT WAVE, PRESSURE AMPLITUDE=*name*
solid or fluid surface name, *reference magnitude*

Abaqus/CAE Usage: The alternative incident wave loading interface is not supported in Abaqus/CAE.

Specifying the Reflection Effects (Alternative Interface)

Conceptually, the alternative interface is the same as the preferred interface; however, the usages are different. For conceptual information, see *Specifying the Reflection Effects*.

Input File Usage: Use the following option in conjunction with the **INCIDENT WAVE* option to define an incident wave reflection plane:

**INCIDENT WAVE REFLECTION*

Abaqus/CAE Usage: The alternative incident wave loading interface is not supported in Abaqus/CAE.

Modeling Incident Wave Loading on a Moving Structure (Alternative Interface)

To model the effect of rigid motion of a structure such as a ship during the incident wave loading history, the standoff point can have a specified velocity. It is assumed that the entire fluid-solid model is moving at this velocity with respect to the source point during the loading and that the speed of the model's motion is low compared to the speed of propagation of the incident wave.

Input File Usage: **INCIDENT WAVE PROPERTY*, NAME=*wave property name*
data line to specify the velocity of the standoff point

Abaqus/CAE Usage: The alternative incident wave loading interface is not supported in Abaqus/CAE.

Example: Submarine Close to the Free Surface

The problem shown in *Figure 4* has the following features: a free surface A_0, seabed A_{sb} as a reflection plane, a wet solid surface A_{sw}, the fluid surface A_{fw} that is tied to the solid surface A_{sw}, and the boundary A_{inf} of the finite modeled domain separating the infinite acoustic medium. The source *S* of the underwater explosion loading is also shown.

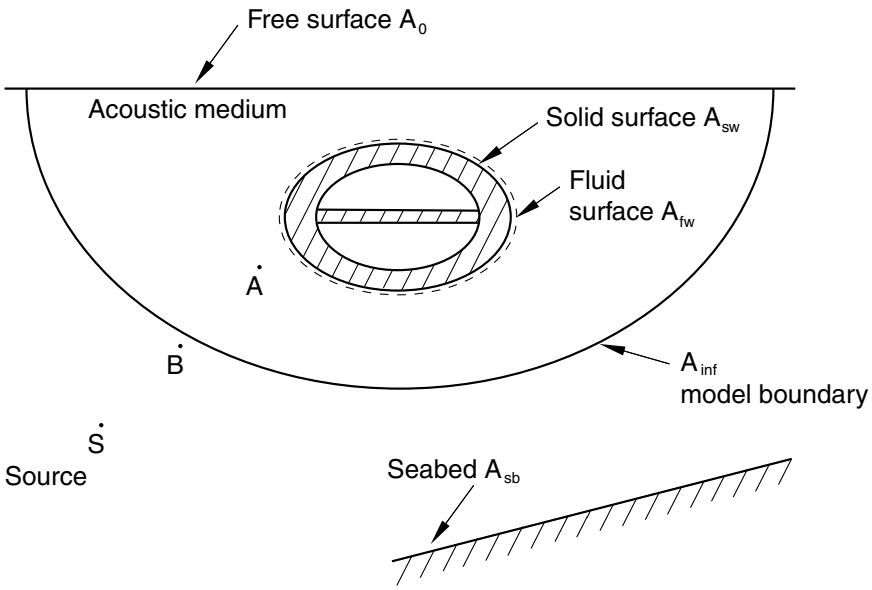


Figure 4: Incident wave loading on a submarine lying near a free surface.

Scattered Wave Solution

Here the scattered wave response in the acoustic medium is of interest along with that of the structure to the incident wave loading. Cavitation in the fluid is not considered in a scattered wave formulation. Similarly, the initial hydrostatic pressure in the fluid is not modeled.

The zero dynamic acoustic pressure boundary condition on the free surface requires both a “soft” reflection plane coinciding with the free surface A_0 and a zero scattered pressure boundary condition at the nodes on this free surface. The incident wave loading is applied on the fluid surface, A_{fw}, and on the wet solid surface, A_{sw}. The incident wave loading can be only of pressure amplitude type since the loading includes a solid surface.

A good location for the standoff node is marked as *A* in *Figure 4*. This node is in the fluid, close to the structure, and closer to incident wave source *S* than any portion of the seabed or the free surface. The standoff node's offset from the loaded surfaces is exaggerated for emphasis in the figure.

The radiation condition is specified on the acoustic surface A_{inf} such that the scattered wave impinging on this boundary with the infinite medium does not reflect back into the computational domain. The seabed is modeled with an incident wave reflection plane on surface A_{sb}. The reflection loss at this seabed surface is modeled using an impedance property.

If the response of the structure in the nonlinear regime is of interest, the initial stress state in the structure should be established using Abaqus/Standard in a static analysis. The stress state in the structure is then imported into Abaqus/Explicit, and the loading on the solid surfaces causing the initial stress state is respecified in the acoustic analysis.

The following template schematically shows some of the Abaqus input file options that are used to solve this problem using the scattered wave formulation:

```
*HEADING
...
*SURFACE, NAME=Afw
Data lines to define the acoustic surface that is wetting the solid
*SURFACE, NAME=Asw
Data lines to define the solid surface that is wetted by the fluid
```

```
*SURFACE, NAME=A_inf
Data lines to define the acoustic surface separating the modeled region from the infinite medium
*INCIDENT WAVE INTERACTION PROPERTY, NAME=IWPROP
*AMPLITUDE, DEFINITION=TABULAR, NAME=PRESSUREVTIME
*TIE, NAME=COUPLING
A_fw, A_sw
*STEP
** For an Abaqus/Standard analysis:
*DYNAMIC
** For an Abaqus/Explicit analysis:
*DYNAMIC, EXPLICIT
** Load the acoustic surface
*INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=PRESSUREVTIME,
PROPERTY=IWPROP
A_fw, source node, standoff node, reference magnitude
*INCIDENT WAVE REFLECTION
Data lines for the reflection plane over the seabed A_sb, seabed_Q
*INCIDENT WAVE REFLECTION
Data lines for a "soft" reflection plane over the free surface A_0.
** Load the solid surface
*INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=PRESSUREVTIME,
PROPERTY=IWPROP
A_sw, source node, standoff node, reference magnitude
*INCIDENT WAVE REFLECTION
Data lines for the reflection plane over the seabed A_sb, seabed_Q
*INCIDENT WAVE REFLECTION
Data lines for a "soft" reflection plane over the free surface A_0.
*BOUNDARY
** zero pressure boundary condition on the free surface
Set of nodes on the free surface A_0, 8, 8, 0.0
*SIMPEDANCE
A_inf,
*END STEP
```

Total Wave Solution

Here, the total wave response in the acoustic medium is of interest along with that of the structure to the incident wave loading. Cavitation in the fluid might be included. Similarly, a linearly varying initial hydrostatic pressure in the fluid can be specified.

The zero dynamic acoustic pressure boundary condition on the free surfaces requires only a zero pressure boundary condition at the nodes on this free surface. A reflection plane should not be included along the free surface. The incident wave loading is applied only on the fluid surface, A_{inf}, that separates the modeled region from the surrounding infinite acoustic medium. No incident wave should be applied directly on the structure surfaces. If the incident wave is considered planar, an acceleration-type amplitude can be used with the incident wave loading. Otherwise, a pressure-type amplitude must be used with the incident wave loading.

An ideal location for the standoff node depends on the type of amplitude used for the time history of the incident wave loading. The location *A* shown in *Figure 4* can be used if the incident wave loading time history is of pressure amplitude type. Otherwise, the location *B* that is just on the boundary A_{inf} and closer to the source *S* than any part of either the seabed or the free surface can be used.

The nonreflecting impedance condition is specified on the acoustic surface, A_{inf}, such that the scattered part of the total wave impinging on this boundary with the infinite medium does not reflect back into the computational domain. The seabed is modeled with an incident wave reflection plane on the surface A_{sb}.

If the response of the structure in the nonlinear regime is of interest, the initial stress state in the structure should be established using Abaqus/Standard in a static analysis. The stress state in the structure is then imported into Abaqus/Explicit, and the loading on the solid surfaces causing the initial stress state is respecified in the acoustic analysis.

The following template schematically shows some of the input file options that are used to solve this problem using the total wave formulation:

```
*HEADING
…
*ACOUSTIC WAVE FORMULATION, TYPE=TOTAL WAVE
*MATERIAL, NAME=CAVITATING_FLUID
*ACOUSTIC MEDIUM, BULK MODULUS
Data lines to define the fluid bulk modulus
*ACOUSTIC MEDIUM, CAVITATION LIMIT
Data lines to define the fluid cavitation limit
…
*SURFACE, NAME=A_fw
Data lines to define the acoustic surface that is wetting the solid
*SURFACE, NAME=A_sw
Data lines to define the solid surface that is wetted by the fluid
*SURFACE, NAME=A_inf
Data lines to define the acoustic surface separating the modeled region from the infinite medium
*INCIDENT WAVE INTERACTION PROPERTY, NAME=IWPROP
*AMPLITUDE, DEFINITION=TABULAR, NAME=PRESSUREVTIME
Data lines to define the pressure-time history at the standoff point
*TIE, NAME=COUPLING
A_fw, A_sw
*INITIAL CONDITIONS, TYPE=ACOUSTIC STATIC PRESSURE
Data lines to define the initial linear hydrostatic pressure in the fluid
*STEP
*DYNAMIC, EXPLICIT
** Load the acoustic surface
*INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=PRESSUREVTIME,
PROPERTY=IWPROP
A_inf, source node, standoff node, reference magnitude
*INCIDENT WAVE REFLECTION
Data lines for the reflection plane over the seabed A_sb, seabed_Q
*BOUNDARY
** zero pressure boundary condition on the free surface
Set of nodes on the free surface A_0, 8, 8, 0.0
*SIMPEDANCE
A_inf,
*END STEP
```

Example: Submarine in Deep Water

This problem is similar to the previous example of a submarine close to the free surface except for the following differences. There is no free surface in this problem; and the fluid surface, A_{inf}, and the fluid medium completely enclose the structure. If the structure is sufficiently deep in the water, hydrostatic pressure might be considered

uniform instead of varying linearly with depth. Under this assumption, the initial stress state in the structure can be established with a uniform pressure loading all around it, if desired. In addition, if the structure is sufficiently deep in the water, the hydrostatic pressure might be significant compared to the incident wave loading; hence, the cavitation in the fluid might not be of concern.

Example: Surface Ship

Here the effect of underwater explosion loading on a surface ship is of interest (see *Figure 5*).

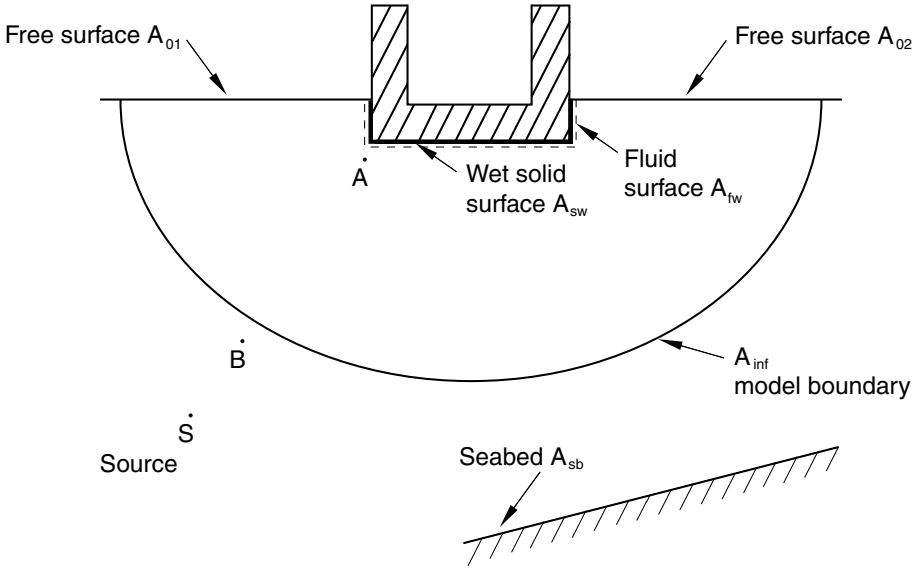


Figure 5: Modeling of incident wave loading on a surface ship.

This problem is similar to the previous example of a submarine close to the free surface except for the following differences. The free surface of fluid is not continuous, and a part of the structure is exposed to the atmosphere. A soft reflection plane coinciding with the free surface is not used in this problem as in the submarine problems under the scattered wave formulation. To be able to use the scattered wave formulation in this case, the modeling technique is used in which the free surface is replaced with “structural fluid” elements. A layer of fluid at the free surface is modeled using nonacoustic elements such as membrane elements. These elements are coupled to the underlying acoustic fluid using a mesh tie constraint. The nonacoustic elements have properties similar to the fluid itself since these elements are replacing the fluid medium near the free surface and should have a thickness similar to the height of the adjacent acoustic elements. Incident wave loading with the scattered wave formulation must now be applied on these newly created surfaces as well. This technique has the added advantage of providing the deformed shape of the free surface under the loading.

The following template shows some of the Abaqus input file options used for this case:

```
*HEADING
…
*SURFACE, NAME=A01_structuralfluid
Data lines to define the "structural fluid" surface
*SURFACE, NAME=A01_acousticfluid
Data lines to define the adjacent acoustic fluid surface
*SURFACE, NAME=A02_structuralfluid
Data lines to define the "structural fluid" surface
*SURFACE, NAME=A02_acousticfluid
Data lines to define the adjacent acoustic fluid surface
*SURFACE, NAME=Asw_solid
Data lines to define the actual solid surface that is wetted by the fluid
*SURFACE, NAME=Asw_fluid
```

Data lines to define the actual acoustic surface that is adjacent to the structure

**SURFACE*, NAME=A_{inf}
Data lines to define the acoustic surface separating the modeled region from the infinite medium
**INCIDENT WAVE INTERACTION PROPERTY*, NAME=IWPROP
**AMPLITUDE*, DEFINITION=TABULAR, NAME=PRESSUREVTIME
Data lines to define the pressure-time history at the standoff point
**TIE*, NAME=COUPLING
Asw_fluid, Asw_solid
A01_acousticfluid, A01_structuralfluid
A02_acousticfluid, A02_structuralfluid
**STEP*
** For an Abaqus/Standard analysis:
**DYNAMIC*
** For an Abaqus/Explicit analysis:
**DYNAMIC*, EXPLICIT
** Load the acoustic surfaces
**INCIDENT WAVE INTERACTION*, PRESSURE AMPLITUDE=PRESSUREVTIME,
PROPERTY=IWPROP
A01_acousticfluid, source point, standoff point, reference magnitude
**INCIDENT WAVE REFLECTION*

Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
**INCIDENT WAVE INTERACTION*, PRESSURE AMPLITUDE=PRESSUREVTIME,
PROPERTY=IWPROP
A02_acousticfluid, source point, standoff point, reference magnitude
**INCIDENT WAVE REFLECTION*

Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
**INCIDENT WAVE INTERACTION*, PRESSURE AMPLITUDE=PRESSUREVTIME,
PROPERTY=IWPROP
Asw_fluid, source point, standoff point, reference magnitude
**INCIDENT WAVE REFLECTION*

Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
** Load the solid surfaces
**INCIDENT WAVE INTERACTION*, PRESSURE AMPLITUDE=PRESSUREVTIME,
PROPERTY=IWPROP
A01_structuralfluid, source point, standoff point, reference magnitude
**INCIDENT WAVE REFLECTION*

Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
**INCIDENT WAVE INTERACTION*, PRESSURE AMPLITUDE=PRESSUREVTIME,
PROPERTY=IWPROP
A02_structuralfluid, source point, standoff point, reference magnitude
**INCIDENT WAVE REFLECTION*

Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
**INCIDENT WAVE INTERACTION*, PRESSURE AMPLITUDE=PRESSUREVTIME,
PROPERTY=IWPROP
Asw_solid, source point, standoff point, reference magnitude
**INCIDENT WAVE REFLECTION*

Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
**SIMPEDANCE*

A_{inf},
**END STEP*

Compared to the total wave formulation analysis of a submarine close to the free surface, the following differences are noteworthy. As shown in *Figure 5*, the free surface with zero dynamic pressure boundary condition is now split into two parts: A_{01} and A_{02}. The fluid surface wetting the ship (A_{fw}) and the wetted ship surface (A_{sw}),

which are tied together, do not encircle the whole structure. Besides these differences, the modeling considerations for the surface ship problem are similar to the total wave analysis of the submarine near the free surface.

Example: Airblast Loading on a Structure

Here the effect of airblast (explosion in the air) loading on a structure is of interest (see *Figure 6*).

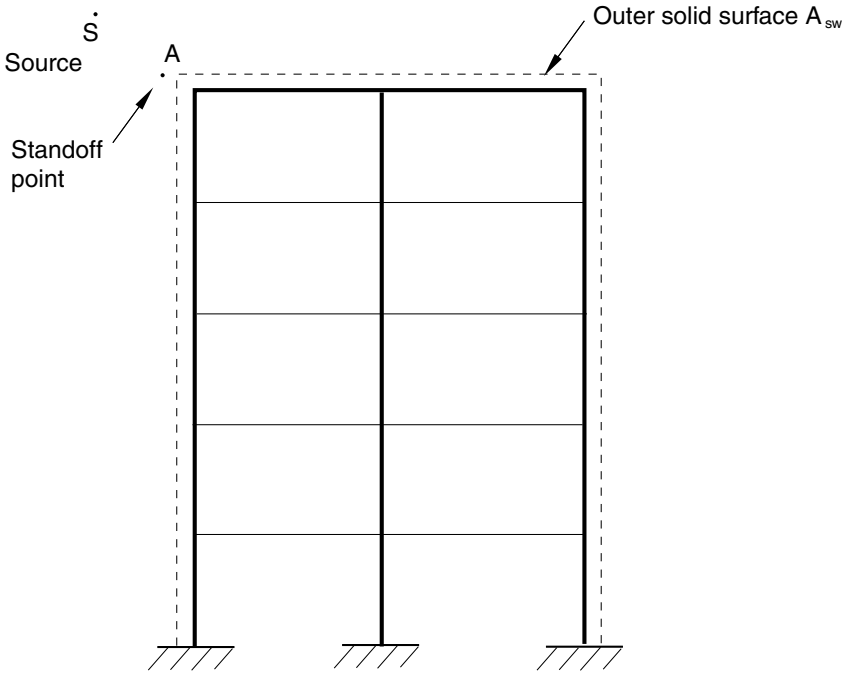


Figure 6: Modeling of airblast loading on a structure.

Since the stiffness and inertia of the air medium are negligible, the acoustic medium is not modeled. Rather the incident wave loading is applied directly on the structure itself. The solid surface A_{sw} where the incident wave loading is applied is shown in *Figure 6*. Since the acoustic medium is not modeled, the total wave and the scattered wave formulations are identical.

Example: Fluid Cavitation without Incident Wave Loading

You might be interested in modeling acoustic problems in Abaqus/Explicit where the loading is applied through either prescribed pressure boundaries or specified pressure-conjugate concentrated loads. Choice of the scattered or the total wave formulation is not relevant in these problems even when the acoustic medium is capable of cavitation.

Pore Fluid Flow

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Loads*
- **CFLOW*
- **DFLOW*
- **DSFLOW*
- **FLOW*
- **SFLOW*
- *Defining a surface pore fluid flow*
- *Defining a concentrated pore fluid flow*

Overview

You can prescribe pore fluid flow in coupled pore fluid diffusion/stress analyses and in the geostatic stress field procedure.
For more information, see *Coupled Pore Fluid Diffusion and Stress Analysis* and *Geostatic Stress State*.

Pore fluid flow can be prescribed by:

- defining seepage coefficients and sink pore pressures on element faces or surfaces;
- defining drainage-only seepage coefficients on element faces or surfaces that are applied only when surface pore pressures are positive; or
- prescribing an outward normal flow velocity directly at nodes, on element faces, or on surfaces.

Defining Pore Fluid Flow as a Function of the Current Pore Pressure in Consolidation Analysis

In consolidation analysis you can provide seepage coefficients and sink pore pressures on element faces or surfaces to control normal pore fluid flow from the interior of the region modeled to the exterior of the region.

The surface condition assumes that the pore fluid flows in proportion to the difference between the current pore pressure on the surface, u_w, and some reference value of pore pressure, u_w^∞:

$$v_n = k_s \left(u_w - u_w^\infty \right),$$

where

v_n

is the component of the effective velocity of the pore fluid in the direction of the outward normal to the surface;

k_s

is the seepage coefficient;

u_w

is the current pore pressure at this point on the surface; and

u_w^∞

is a reference pore pressure value.

Specifying Element-Based Pore Fluid Flow

To define element-based pore fluid flow, specify the element or element set name; the distributed load type; the reference pore pressure, u_w^∞; and the reference seepage coefficient, k_s. The face of the elements upon which the normal flow is enforced is identified by a seepage distributed load type. The seepage types available depend on the element type (see *About the Element Library*).

Input File Usage: **FLOW*
element number or element set name, Q*n*, u_w^∞, k_s

Abaqus/CAE Usage: Pore fluid flow cannot be defined as a function of the current pore pressure in Abaqus/CAE.

Specifying Surface-Based Pore Fluid Flow

To define surface-based pore fluid flow, specify a surface name, the seepage flow type, the reference pore pressure, and the reference seepage coefficient. The element-based surface (see *Element-Based Surface Definition*) contains the element and face information.

Input File Usage: **SFLOW*
surface name, Q, u_w^∞, k_s

Abaqus/CAE Usage: Pore fluid flow cannot be defined as a function of the current pore pressure in Abaqus/CAE.

Defining Drainage-Only Flow

Drainage-only flow types can be specified for element-based or surface-based pore fluid flow to indicate that normal pore fluid flow occurs only from the interior to the exterior region of the model. The drainage-only flow surface condition assumes that the pore fluid flows in proportion to the magnitude of the current pore pressure on the surface, u_w, when that pressure is positive:

$$v_n = k_s u_w, \qquad u_w > 0$$
$$v_n = 0 \qquad\quad u_w \le 0,$$

where

v_n

is the component of the pore fluid velocity in the direction of the outward normal to the surface;

k_s

is the seepage coefficient; and

u_w

is the current pore pressure at this point on the surface.

Figure 1 illustrates this pore pressure–velocity relationship. This surface condition is designed for use with the total pore pressure formulation (see *Coupled Pore Fluid Diffusion and Stress Analysis*), mainly for cases where the phreatic surface intersects an exterior surface that is free to drain. See *Calculation of phreatic surface in an earth dam* for an example of this type of calculation.

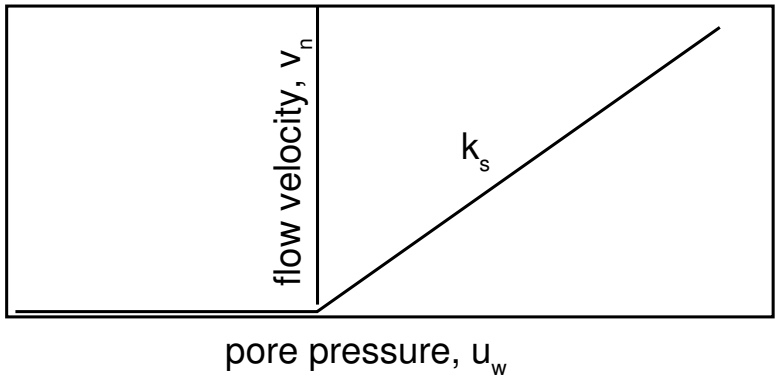


Figure 1: Drainage-only pore pressure–velocity relationship.

When surface pore pressures are negative, the constraint will properly enforce the condition that no fluid can enter the interior region. When surface pore pressures are positive, the constraint will permit fluid flow from the interior to the exterior region of the model. When the seepage coefficient value, k_s, is large, this flow will approximately enforce the requirement that the pore pressure should be zero on a freely draining surface. To achieve this condition, it is necessary to choose the value of k_s to be much larger than a characteristic seepage coefficient for the material in the underlying elements:

$$k_s \gg k/\gamma_w c,$$

where

k

is the permeability of the underlying material;

γ_w

is the fluid specific weight; and

c

is a characteristic length of the underlying elements.

Values of $k_s \approx 10^5 k/\gamma_w c$ will be adequate for most analyses. Larger values of k_s could result in poor conditioning of the model. In all cases the freely draining flow type represents discontinuously nonlinear behavior, and its use might require appropriate solution controls (see *Commonly Used Control Parameters*).

Input File Usage: Use the following option to define element-based drainage-only flow:

**FLOW*
element number or element set name, Q*n*D, k_s

Use the following option to define surface-based drainage-only flow:

**SFLOW*
surface name, QD, k_s

Abaqus/CAE Usage: Pore fluid flow cannot be defined as a function of the current pore pressure in Abaqus/CAE.

Modifying or Removing Seepage Coefficients and Reference Pore Pressures

Seepage coefficients and reference pore pressures can be added, modified, or removed as described in *About Loads*.

Specifying a Time-Dependent Reference Pore Pressure

The magnitude of the reference pore pressure, u_w^∞, can be controlled by referring to an amplitude curve. If different variations are needed for different portions of the flow, repeat the flow definition with each referring to its own amplitude curve. See *About Loads* and *Amplitude Curves* for details.

Defining Nonuniform Flow in a User Subroutine

To define nonuniform flow, the variation of the reference pore pressure and the seepage coefficient as functions of position, time, pore pressure, etc. can be defined in user subroutine *FLOW*.

Input File Usage: Use the following option to define a nonuniform element-based flow:

**FLOW*
element number or element set name, Q*n*NU

Use the following option to define a nonuniform surface-based flow:

**SFLOW*
surface name, QNU

Abaqus/CAE Usage: User subroutine *FLOW* is not supported in Abaqus/CAE.

Prescribing Seepage Flow Velocity and Seepage Flow Directly in Consolidation Analysis

You can directly prescribe an outward normal flow velocity, v_n, across a surface or an outward normal flow at a node in consolidation analysis.

Prescribing Element-Based Seepage Flow Velocity

To prescribe an element-based seepage flow velocity, specify the element or element set name, the seepage type, and the outward normal flow velocity. The face of the element for which the seepage flow is being defined is identified by the seepage type. The seepage types available depend on the element type (see *About the Element Library*).

Input File Usage: **DFLOW*
element number or element set name, S*n*, v_n

Abaqus/CAE Usage: Load module: **Create Load**: choose **Fluid** for the **Category** and **Surface pore fluid** for the **Types for Selected Step**: select region: **Distribution**: select an analytical field, **Magnitude**: v_n

Prescribing Surface-Based Seepage Flow Velocity

To prescribe a surface-based seepage flow velocity, specify a surface name, the seepage flow type, and the pore fluid velocity. The element-based surface (see *Element-Based Surface Definition*) contains the element and face information.

Input File Usage: **DSFLOW*
surface name, S, v_n

Abaqus/CAE Usage: Load module: **Create Load**: choose **Fluid** for the **Category** and **Surface pore fluid** for the **Types for Selected Step**: select region: **Distribution**: **Uniform**, **Magnitude**: v_n

Prescribing Node-Based Seepage Flow

To prescribe node-based seepage flow, specify the node or node set name and the magnitude of the flow per unit time.

Input File Usage: **CFLOW*
node number or node set name, , *magnitude*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Fluid** for the **Category** and **Concentrated pore fluid** for the **Types for Selected Step**: select region: **Magnitude**: *magnitude*

Prescribing Seepage Flow at Phantom Nodes for Enriched Elements

For an enriched element (see *Modeling Discontinuities as an Enriched Feature Using the Extended Finite Element Method*), you can specify the seepage flow at a phantom node that is originally located coincident with the specified real node.

Alternatively, you can specify the seepage flow at a phantom node located at an element edge between two specified real corner nodes directly or indicate that the pore pressure applied to a phantom node located at an element edge is interpolated from the specified real corner nodes.

Input File Usage: Use the following option to specify the seepage flow at a phantom node originally located coincident with the specified real node:

**CFLOW*, PHANTOM=NODE
node number, , *magnitude*

Use the following option to specify the seepage flow at a phantom node located at an element edge:

**CFLOW*, PHANTOM=EDGE
first corner node number, *second corner node number*, *magnitude*

Use the following option to indicate that the pore pressure applied to a phantom node located at an element edge will be interpolated automatically from the specified real corner nodes when the enriched element is cracked:

**CFLOW*, PHANTOM=INCLUDED
node or node set name, , *magnitude*

Abaqus/CAE Usage: Prescribing seepage flow at phantom nodes for enriched elements is not supported in Abaqus/CAE

Modifying or Removing Seepage Flow Velocities and Seepage Flow

Seepage flow velocities can be added, modified, or removed as described in *About Loads*.

Specifying Time-Dependent Flow Velocity and Flow

The magnitude of the seepage velocity, v_n, can be controlled by referring to an amplitude curve. To specify different variations for different flows, repeat the seepage flow velocity or seepage flow definition with each referring to its own amplitude curve. See *About Loads* and *Amplitude Curves* for details.

Defining Nonuniform Flow Velocities in a User Subroutine

To define nonuniform element-based or surface-based flow, the variation of the seepage magnitude as a function of position, time, pore pressure, etc. can be defined in user subroutine *DFLOW*. If the optional seepage velocity, v_n, is specified directly, this value is passed into user subroutine *DFLOW* in the variable used to define the seepage magnitude.

Input File Usage: Use the following option to define nonuniform element-based flow:

**DFLOW*
element number or element set name, S*n*NU, v_n

Use the following option to define nonuniform surface-based flow:

**DSFLOW*
surface name, SNU, v_n

Abaqus/CAE Usage: Use the following input to define nonuniform surface-based flow:

Load module: **Create Load**: choose **Fluid** for the **Category** and **Surface pore fluid** for the **Types for Selected Step**: select region: **Distribution: User-defined**, **Magnitude**: v_n

Nonuniform element-based flow is not supported in Abaqus/CAE.

Prescribed Assembly Loads

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Prescribed Conditions*
- **BOUNDARY*
- **CLOAD*
- **PRE-TENSION SECTION*
- **SURFACE*
- *Bolt loads*

Overview

You can define pre-tension sections in Abaqus/Standard to prescribe assembly loads in bolts or any other type of fastener.

Assembly loads:

- can be used to simulate the loading of fasteners in a structure;
- are applied across user-defined pre-tension sections;
- are applied to pre-tension nodes that are associated with the pre-tension sections; and
- require the specification of pre-tension loads or tightening adjustments.

Concept of an Assembly Load

Figure 1 is a simple example that illustrates the concept of an assembly load.

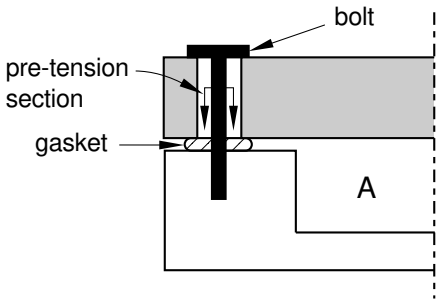


Figure 1: Example of assembly load.

Container *A* is sealed by pre-tensioning the bolts that hold the lid, which places the gasket under pressure. This pre-tensioning is simulated in Abaqus/Standard by adding a “cutting surface,” or pre-tension section, in the bolt, as shown in *Figure 1*, and subjecting it to a tensile load. By modifying the elements on one side of the surface, Abaqus/Standard can automatically adjust the length of the bolt at the pre-tension section to achieve the prescribed

amount of pre-tension. In later steps further length changes can be prevented so that the bolt acts as a standard, deformable component responding to other loadings on the assembly.

Modeling an Assembly Load

Abaqus/Standard allows you to prescribe assembly loads across fasteners that are modeled by continuum, truss, or beam elements. The steps needed to model an assembly load vary slightly depending on the type of elements used to model the fasteners.

Modeling a Fastener with Continuum Elements

In continuum elements the pre-tension section is defined as a surface inside the fastener that "cuts" it into two parts (see *Figure 2*). The pre-tension section can be a group of surfaces for cases where a fastener is composed of several segments.

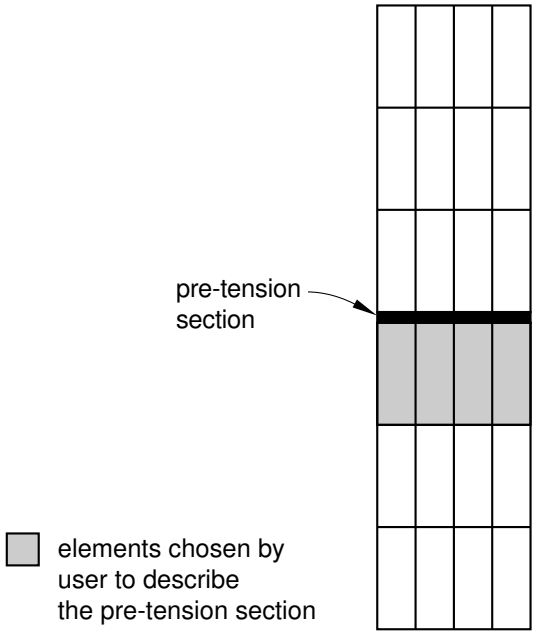


Figure 2: Pre-tension section defined using continuum elements.

The element-based surface contains the element and face information (see *Element-Based Surface Definition*). You must convert the surface into a pre-tension section across which pre-tension loads can be applied and assign a controlling node to the pre-tension section.

Input File Usage: Use the following options to model an assembly load across a fastener that is modeled with continuum elements:

**SURFACE*, TYPE=ELEMENT, NAME=*surface_name*
**PRE-TENSION SECTION*, SURFACE=*surface_name*, NODE=*n*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Bolt load** for the **Types for Selected Step**

Assigning a Controlling Node to the Pre-Tension Section

The assembly load is transmitted across the pre-tension section by means of the pre-tension node. The pre-tension node should not be attached to any element in the model. It has only one degree of freedom (degree of freedom 1), which represents the relative displacement at the two sides of the cut in the direction of the normal (see *Figure 3*). The coordinates of this node are not important.

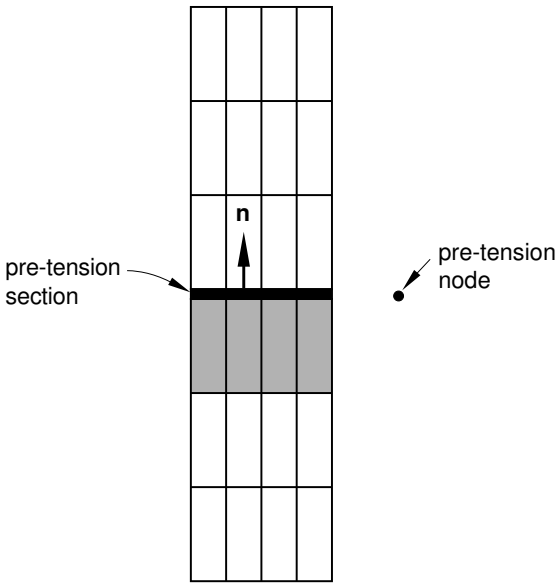


Figure 3: Normal to the pre-tension section; this normal should face away from the underlying elements.

Defining the Normal to the Pre-Tension Section

Abaqus/Standard computes an average normal to the section—in the positive surface direction, facing away from the continuum elements used to generate the surface—to determine the direction along which the pre-tension is applied. You may also specify the normal directly (when the desired direction of loading is different from the average normal to the pre-tension section). You can specify if the normal is updated or fixed when performing a large-displacement analysis (see *Updating the Normal in a Large-Displacement Analysis*).

Recognizing Elements on Either Side of the Pre-Tension Section

For all the elements that are connected to the pre-tension section by at least one node, Abaqus/Standard must determine on which side of the pre-tension section each element is located. This process is crucial for the prescribed assembly load to work properly.

The elements used to define the section are referred to as “base elements” in this discussion. All elements on the same side of the section as the base elements are referred to as the “underlying elements.” All elements connected to the section that share faces (or in two-dimensional problems, edges) with the base elements are added to the list of underlying elements. This is a repetitive process that enables Abaqus/Standard to find the underlying elements in almost all meshes—triangles; wedges; tetrahedra; and embedded beams, trusses, shells, and membranes—that were not used in the definition of the surface (see *Figure 4*).

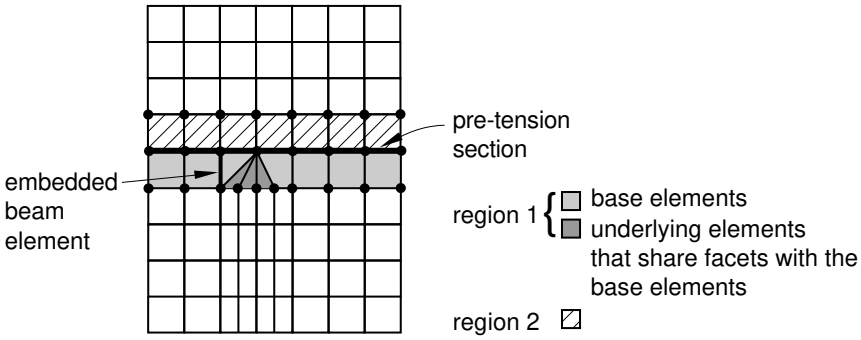


Figure 4: The base elements are used to find the underlying elements.

In most cases this process will group all of the elements that are connected to the section into two regions, as shown in the figure. In rare instances this process may group the elements in more than two regions, in particular if line elements cross over element boundaries. An example is shown in *Figure 5*; it has three regions, where region 1 is the underlying region.

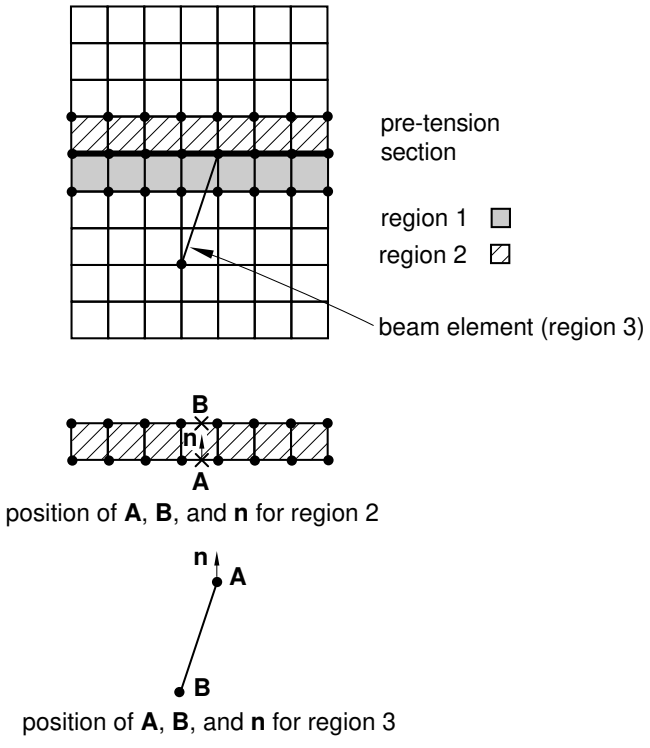


Figure 5: An additional underlying element is found.

For each region other than region 1 an additional step is necessary to determine on which side of the section the region is located. Abaqus/Standard computes an average normal, $\mathbf{n}$, for all the nodes of the region that belong to the section; it also computes an average position ($\mathbf{A}$) of all these nodes. In addition, it computes an average position ($\mathbf{B}$) of the remaining nodes of the region. If the dot product between the normal $\mathbf{n}$ and the vector $\boldsymbol{AB}$ is negative, the region is assumed to be an underlying region and is added to region 1. This additional step is illustrated in *Figure 5* for regions 2 and 3.

This additional step produces an incorrect separation for the beam element shown in *Figure 6* since the beam is not found to be an underlying element.

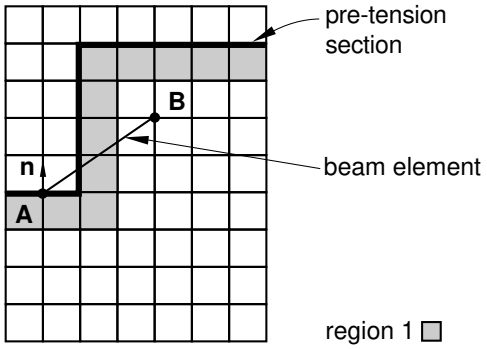


Figure 6: An additional underlying element is not found.

If the pre-tension section has an odd shape and one or more line elements that cross over element boundaries are connected to it, consult the list of the underlying elements given in the data (`.dat`) file to make sure that the underlying elements are listed correctly.

Elements that are connected only to the nodes on the pre-tension section, including single-node elements (such as SPRING1, DASHPOT1, and MASS elements) are not included as underlying elements: they are considered to be attached to the other side of the section.

Modeling a Fastener with Truss or Beam Elements

When a pre-tensioned component is modeled with truss or beam elements, the pre-tension section is reduced to a point. The section is assumed to be located at the last node of the element as defined by the element connectivity (see *Beam Element Library* and *Truss Element Library* for a definition of the node ordering for beam and truss elements, respectively), with its normal along the element directed from the first to the last node. As a result, the section is defined entirely by just specifying the element to which an assembly load must be prescribed and associating it with a pre-tension node.

As in the case of a surface-based pre-tension section, the node has only one degree of freedom (degree of freedom 1), which represents the relative displacement on the two sides of the cut in the direction of the normal (see *Figure 7*). The coordinates of the node are not important.

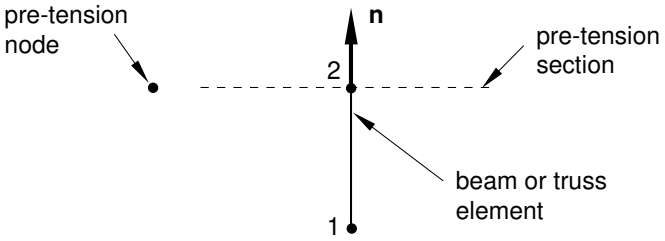


Figure 7: Pre-tension section defined using a truss or beam element.

Abaqus/Standard computes the normal as the vector from the first to the last node in the connectivity of the underlying element. Alternatively, you can specify the normal to the section directly. You can specify if the normal is updated or fixed when performing a large-displacement analysis (see *Updating the Normal in a Large-Displacement Analysis*).

Input File Usage: Use the following option to model an assembly load across fasteners modeled with beam or truss elements:

**PRE-TENSION SECTION*, ELEMENT=*element_number*, NODE=*n*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Bolt load** for the **Types for Selected Step**

Updating the Normal in a Large-Displacement Analysis

You can specify if the normal is updated or fixed when performing a large-displacement analysis.

Input File Usage: Use the following option to indicate that the normal is updated when performing a large-displacement analysis:

**PRE-TENSION SECTION*, FOLLOWER=YES

Use the following option to indicate that the normal is fixed during the analysis:

**PRE-TENSION SECTION*, FOLLOWER=NO

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Bolt load** for the **Types for Selected Step**: toggle on **Varying normal for pre-tension section**

Defining Multiple Pre-Tension Sections

You can define multiple pre-tension sections by repeating the pre-tension section definition input. Each pre-tension section should have its own pre-tension node.

Use with Nodal Transformations

A local coordinate system (see *Transformed Coordinate Systems*) cannot be used at a pre-tension node. It can be used at nodes located on pre-tension sections.

Applying the Prescribed Assembly Load

The pre-tension load is transmitted across the pre-tension section by means of the pre-tension node.

Prescribing the Pre-Tension Force

You can apply a concentrated load to the pre-tension node. This load is the self-equilibrating force carried across the pre-tension section, acting in the direction of the normal on the part of the fastener underlying the pre-tension section (the part that contains the elements that were used in the definition of the pre-tension section; see *Figure 8*).

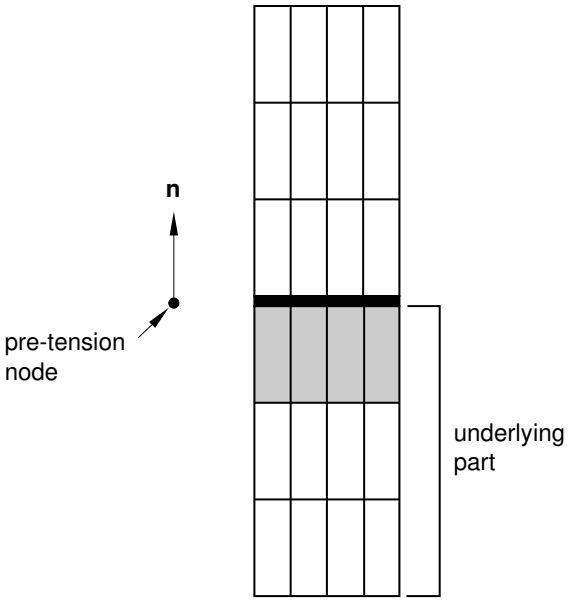


Figure 8: The prescribed assembly load is given at the pre-tension node and applied in direction n.

Input File Usage: **CLOAD*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Bolt load** for the **Types for Selected Step**: toggle on **Varying normal for pre-tension section**

Prescribing a Tightening Adjustment

You can prescribe a tightening adjustment of the pre-tension section by using a nonzero boundary condition at the pre-tension node (which corresponds to a prescribed change in the length of the component cut by the pre-tension section in the direction of the normal).

Input File Usage: **BOUNDARY*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Bolt load** for the **Types for Selected Step**: select surface and, if necessary, bolt axis: **Method: Adjust length**

Controlling the Pre-Tension Node during the Analysis

You can maintain the initial adjustment of the pre-tension section by using a boundary condition fixing the degrees of freedom at their current values at the start of the step once an initial pre-tension is applied in the fastener. This technique enables the load across the pre-tension section to change according to the externally applied loads to maintain equilibrium. If the initial adjustment of a section is not maintained, the force in the fastener remains constant.

When a pre-tension node is not controlled by a boundary condition, make sure that the components of the structure are kinematically constrained; otherwise, the structure could fall apart due to the presence of rigid body modes. Abaqus/Standard issues a warning message if it does not find any boundary condition or load on a pre-tension node during the first step of the analysis.

Display of Results

Abaqus/Standard automatically adjusts the length of the component at the pre-tension section to achieve the prescribed amount of pre-tension. This adjustment is done by moving the nodes of the underlying elements that lie on the pre-tension section relative to the same nodes when they appear in the other elements connected to the pre-tension section. As a result, the underlying elements will appear shrunk, even though they carry tensile stresses when a pre-tension is applied.

Limitations When Using Assembly Loads

Assembly loads are subject to the following limitations:

- An assembly load cannot be specified within a substructure.
- If a submodeling analysis is performed (*About Submodeling*), any pre-tension section should not cross regions where driven nodes are specified. In other words, a pre-tension section should appear either entirely in the region of the global model that is not part of a submodel or entirely in the region of the global model that is part of a submodel. In the latter case, a pre-tension section must also appear in the submodel when the submodel analysis is performed.
- Nodes of a pre-tension section should not be connected to other parts of the body through multi-point constraints (*General Multi-Point Constraints*). These nodes can be connected to other parts of the body through equations (*Linear Constraint Equations*). However, an equation connecting a node on the pre-tension section to a node located on the underlying side of the section introduces a constraint that spans across the pre-tension cut and, therefore, interacts directly with the application of the pre-tension load. On the other hand, an equation connecting a node on the pre-tension section to a node on the other side of the section does not influence the application of the pre-tension load.

Procedures

Any of the Abaqus/Standard procedures that use element types with displacement degrees of freedom can be used. Static analysis is the most likely procedure type to be used when prescribing the initial pre-tension (*Static Stress Analysis*). Other analysis types such as coupled temperature-displacement (*Sequentially Coupled Thermal-Stress Analysis*) or coupled thermal-electrical-structural (*Fully Coupled Thermal-Electrical-Structural Analysis*) can also be used. Once the initial pre-tension is applied, a static or dynamic analysis (*About Dynamic Analysis Procedures*) may, for instance, be used to apply additional loads while maintaining the tightening adjustment.

Output

The total force across the pre-tension section is the sum of the reaction force at the pre-tension node plus any concentrated load specified at that node. The total force across the pre-tension section is available as output using the output variable identifier TF (see *Abaqus/Standard Output Variable Identifiers*). The forces are along the normal direction. The shear force across the pre-tension section is not available for output.

The tightening adjustment of the pre-tension section is available as the displacement of the pre-tension node. The output of displacement is requested using output identifier U. Only the adjustment normal to the pre-tension section is output since there is no adjustment in any other direction.

The stress distribution across the pre-tension section is not available directly; however, the stresses in the underlying elements can be displayed readily. Alternatively, a tied contact pair can be inserted at the location of the pre-tension section to enable stress distribution output by means of output identifiers CPRESS and CSHEAR. See *Defining Tied Contact in Abaqus/Standard* for details on defining tied contact.

Input File Template

```
*HEADING
Prescribed assembly load; example using continuum elements
…
*NODE
Optionally define the pre-tension node
*SURFACE, NAME=name
Data lines that specify the elements and their associated faces to define the pre-tension section
*PRE-TENSION SECTION, SURFACE=name, NODE=pre-tension_node
**
*STEP
** Application of the pre-tension across the section
*STATIC
Data line to control time incrementation
*CLOAD
pre-tension_node, 1, pre-tension_value
or
*BOUNDARY,AMPLITUDE=amplitude
pre-tension_node, 1, 1, tightening adjustment
*END STEP
*STEP
** maintain the tightening adjustment and apply new loads
*STATIC or *DYNAMIC
Data line to control time incrementation
*BOUNDARY,FIXED
pre-tension_node, 1, 1
*BOUNDARY
Data lines to prescribe other boundary conditions
*CLOAD or *DLOAD
Data lines to prescribe other loading conditions
…
*END STEP
```

Predefined Fields

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Prescribed Conditions*
- *File Output Format*
- **TEMPERATURE*
- **FIELD*
- **PORE FLUID PRESSURE*
- **PRESSURE STRESS*
- **MASS FLOW RATE*
- *Defining a temperature field*
- *Defining values for predefined field variables*

Overview

Predefined fields are time-dependent, non-solution-dependent fields that exist over the spatial domain of the model. Temperature is the most commonly defined field.

This section describes how to specify the values of the following types of predefined fields during an analysis:

- temperature,
- field variables, including predefined pore fluid pressure,
- equivalent pressure stress, and
- mass flow rate.

The procedures in which these fields can be used are outlined in *About Prescribed Conditions*.

Temperature, field variables, equivalent pressure stress, and mass flow rate are time-dependent, predefined (not solution-dependent) fields that exist over the spatial domain of the model. They can be defined:

- by entering the data directly,
- by reading an Abaqus results file generated during a previous analysis (usually an Abaqus/Standard heat transfer analysis), or
- in a user subroutine.

Temperature and field variables can also be defined by reading an Abaqus output database file generated during a previous analysis.

Field variables can also be made solution dependent, which allows you to introduce additional nonlinearities in the Abaqus material models.

Predefined Temperature

In stress/displacement analysis the temperature difference between a predefined temperature field and any initial temperatures (*Initial Conditions*) will create thermal strains if a thermal expansion coefficient is given for the material (*Thermal Expansion*). The predefined temperature field also affects temperature-dependent material properties, if any. In Abaqus/Explicit temperature-dependent material properties may cause longer run times than constant properties.

You define the magnitude and time variation of temperature at the nodes, and Abaqus interpolates the temperatures to the material points.

Input File Usage: Use the following option to specify a predefined temperature field:

**TEMPERATURE*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**

Restrictions

Do not specify predefined temperature fields in a pure heat transfer analysis, a coupled thermal-electrical analysis, a fully coupled temperature-displacement analysis, or a fully coupled thermal-electrical-structural analysis; instead, specify a boundary condition (*Boundary Conditions*) to prescribe temperature degrees of freedom (11, 12, ...).

Predefined temperature fields cannot be specified in an adiabatic analysis step or in any mode-based dynamic analysis step.

To specify a predefined temperature field in a restart analysis, the corresponding predefined field must have been specified in the original analysis as either initial temperatures (see *Defining Initial Temperatures*) or a predefined temperature field.

Predefined Field Variables

The usage and treatment of predefined field variables is exactly analogous to that of temperature. You can prescribe the magnitude and time variation of the field at all of the nodes of the model, and Abaqus will interpolate the values to the material points.

When prescribing field variable values, you must specify the field variable number being defined; the default is field variable number 1. Field variables must be numbered consecutively starting from one. Repeat the field variable definition to define more than one field variable.

The field variable can be a real field (such as an electromagnetic field) generated by a previous simulation (Abaqus or another analysis code). It can also be an artificial field that you define to modify certain material properties during the course of an analysis. For example, suppose that you wish to vary Young's modulus linearly between 30×10^6 and 35×10^6 during the response. The linear elastic material definition shown in *Table 1* could be used.

Table 1: Sample material definition.

Number of field variable dependencies: 1		
Young's modulus	**Poisson's ratio**	**Value of field variable 1**
30.E6	0.3	1.0
35.E6	0.3	2.0

Define an initial condition to specify the initial value of field variable 1 as 1.0 for a node set. Then, define a predefined field variable in the analysis step to specify the value of field variable 1 as 2.0 for the node set. Young's modulus will vary smoothly over the course of the step as the field variable's value is ramped from 1.0 to 2.0 at all nodes in the node set.

Field variables can also be used to vary real properties in space by making the properties depend on field variables, as above, and by assigning different field variable values to different nodes.

Making properties depend on field variables will increase the computer time required, since Abaqus must perform the necessary table look-ups.

In an Abaqus/Standard stress/displacement analysis the difference between a predefined field variable and its initial value (*Initial Conditions*) will create volumetric strains analogous to thermal strains if a field expansion coefficient (for the corresponding field variable) is given for the material (*Thermal Expansion*).

Input File Usage: Use the following option to specify a predefined field variable:

**FIELD*, VARIABLE=*n*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Field** for the **Types for Selected Step**; select region; **Field variable number**: *n*

Pore Fluid Pressure

You can apply a known pore fluid pressure field as a predefined field variable in a static or in an explicit dynamic stress analysis. You associate the known pore fluid pressure field with a predefined field variable, *n*. For additional details on making this association, see *Pore Fluid Pressure*.

Input File Usage: Use the following options to specify a predefined field variable:

**MATERIAL*, NAME=*name*
**ELASTIC*
**PORE FLUID PRESSURE*, FIELD=*n*
...
**STEP*
**STATIC* (or **DYNAMIC*, EXPLICIT)
**FIELD*, VARIABLE=*n*
...
**END STEP*

Abaqus/CAE Usage: Specifying a predefined pore fluid pressure is not supported in Abaqus/CAE.

Restrictions

To specify a predefined field variable in a restart analysis, the corresponding predefined field must have been specified in the original analysis as either an initial field variable value (see *Defining Initial Values of Predefined Field Variables*) or a predefined field variable.

Predefined Pressure Stress

You can apply equivalent pressure stress as a predefined field in a mass diffusion analysis. The usage and treatment of pressure stresses is analogous to that of temperatures and field variables. In Abaqus equivalent pressure stresses are positive when they are compressive.

Input File Usage: Use the following option to specify a predefined equivalent pressure stress field:

**PRESSURE STRESS*

Abaqus/CAE Usage: Predefined equivalent pressure stress is not supported in Abaqus/CAE.

Restrictions

Predefined equivalent pressure stress fields can be specified only in a mass diffusion procedure (see *Mass Diffusion Analysis*).

To specify a predefined equivalent pressure stress field in a restart analysis, the corresponding predefined field must have been specified in the original analysis as either initial pressure stresses (see *Defining Initial Pressure Stress in a Mass Diffusion Analysis*) or a predefined equivalent pressure stress field.

Predefined Mass Flow Rate

You can specify the mass flow rate per unit area (or through the entire section for one-dimensional elements) for forced convection/diffusion elements in a heat transfer analysis. The usage and treatment of mass flow rate is analogous to that of temperatures and field variables.

Input File Usage: Use the following option to specify a predefined mass flow rate field:

**MASS FLOW RATE*

Abaqus/CAE Usage: Predefined mass flow rate is not supported in Abaqus/CAE.

Restrictions

A predefined mass flow rate field can be specified only with forced convection/diffusion elements in a heat transfer procedure (see *Uncoupled Heat Transfer Analysis*).

To specify a predefined mass flow rate field in a restart analysis, the corresponding predefined field must have been specified in the original analysis by using either initial mass flow rates (see *Defining Initial Mass Flow Rates in Forced Convection Heat Transfer Elements*) or a predefined mass flow rate field.

Specifying Uniform Predefined Temperatures and Fields

You can assign uniform predefined temperature and field variables to either the entire model or to node sets that you specify. Omit the node number or node set to apply the specified uniform temperature or field variable to all nodes in the model automatically.

You can specify uniform predefined temperatures and field variables with all element types, including beams and shells. However, the definition of the temperatures and field variables must be compatible with the section definition of the element and with adjacent elements, as explained in *Predefined Fields*.

Input File Usage: Use one of the following options:

**TEMPERATURE*, SECTION SPECIFICATION=UNIFORM
**FIELD*, SECTION SPECIFICATION=UNIFORM

Abaqus/CAE Usage: Specifying uniform predefined temperatures and fields is not supported in Abaqus/CAE.

Reading Initial Values of a Field from a User-Specified Results File

An Abaqus/Standard results file can be used to specify initial values of

- temperature (see *Defining Initial Temperatures*);
- field variables (see *Defining Initial Values of Predefined Field Variables*); and
- pressure stress (see *Defining Initial Pressure Stress in a Mass Diffusion Analysis*).

Field variable values must be read from the temperature record (see *Reading Field Values from a User-Specified Results File* below). The part (`.prt`) file from the original analysis is also required when reading data from the results file.

If the zero increment results were requested as output to the Abaqus/Standard results file (see *Obtaining Results at the Beginning of a Step*), you can define initial values of prescribed fields as those existing at the beginning of a step (the zero increment) in the previous heat transfer analysis (field variables and temperatures) or stress/displacement analysis (pressure stress). The `.fil` file extension is optional.

Reading Initial Values of a Temperature Field from a User-Specified Output Database File

An Abaqus/Standard output database file (in ODB or SIM format) can be used to specify initial values of temperature (see *Defining Initial Temperatures*). The part (`.prt`) file from the original analysis is also required when reading data from the output database file. Temperature values can be read between dissimilar meshes, as described in *Interpolating Initial Temperatures for Dissimilar Meshes from a User-Specified Results or Output Database File*.

Initializing Predefined Field Variables from a User-Specified Output Database File in Abaqus/Standard

In Abaqus/Standard nodal values of temperature (NT), normalized concentrations (NNC), and electric potential (EPOT) can be used to initialize predefined fields (see *Defining Initial Values of Predefined Field Variables*). The part (`.prt`) file from the original analysis is also required when reading data from the output database file. The scalar nodal values can be mapped between dissimilar meshes, as described in *Defining Initial Predefined Field Variables by Interpolating Scalar Nodal Output Variables for Dissimilar Meshes from a User-Specified Output Database File*.

Defining Time-Dependent Fields

The prescribed magnitude of a field can vary with time during a step according to an amplitude function. See *About Prescribed Conditions* and *Amplitude Curves* for details.

Input File Usage: Use one of the following options:

**TEMPERATURE*, AMPLITUDE=*amplitude_name*
**FIELD*, AMPLITUDE=*amplitude_name*
**PRESSURE STRESS*, AMPLITUDE=*amplitude_name*
**MASS FLOW RATE*, AMPLITUDE=*amplitude_name*

Abaqus/CAE Usage: In Abaqus/CAE only predefined temperature fields and predefined field variables are available.

Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Temperature** or **Field** for the **Types for Selected Step**: select region: **Distribution: Direct specification** or select an analytical field or a discrete field, **Amplitude:** *amplitude_name*

Field Propagation

By default, all fields defined in the previous general analysis step remain unchanged in the subsequent general step or in subsequent consecutive linear perturbation steps. Fields do not propagate between linear perturbation steps. You define the fields in effect for a given step relative to the preexisting fields. At each new step the existing fields can be modified and additional fields can be specified. If you specify additional values for a field, the definition of the field will be extended to those nodes where it was previously undefined. Alternatively, you can release all previously applied fields of a given type in a step and specify new ones. In this case any fields of that type that are to be retained must be respecified.

Modifying Fields

By default, when you modify existing temperatures, field variables, pressure stresses, or mass flow rates, all existing values of the field remain.

Input File Usage: Use one of the following options to modify an existing field or to specify an additional field:

**TEMPERATURE*, OP=MOD
**FIELD*, OP=MOD
**PRESSURE STRESS*, OP=MOD
**MASS FLOW RATE*, OP=MOD

Abaqus/CAE Usage: In Abaqus/CAE only predefined temperature fields and predefined field variables are available.

Load module: **Create Predefined Field** or **Predefined Field Manager**: **Edit**

Removing Fields

A field that is removed is reset to the value given as an initial condition or to zero if no initial condition was defined. When fields are reset to their initial conditions, the amplitude referred to in the field definition does not apply. In Abaqus/Standard the amplitude variation defined for the step governs the behavior; in most Abaqus/Standard procedures the default is to ramp the fields back to their initial conditions (see *Defining an Analysis*). In Abaqus/Explicit the values are always ramped linearly over the step back to their initial conditions.

If the temperatures, field variables, pressure stresses, or mass flow rates are reset to a new value (not to their initial conditions), the amplitude referred to in the field definition applies.

If you choose to remove any field in a step, no fields of that type will be propagated from the previous general step. All fields of the same type that are in effect during this step must be respecified.

Input File Usage: Use one of the following options to release all previously applied fields of a particular type and to specify new fields:

**TEMPERATURE*, OP=NEW
**FIELD*, OP=NEW
**PRESSURE STRESS*, OP=NEW
**MASS FLOW RATE*, OP=NEW

If the OP=NEW parameter is used on any field option in a step, it must be used on all field options of the same type within the step.

Abaqus/CAE Usage: Use the following option to reset a temperature field to the value prescribed in the initial step (or to zero if no initial value was defined):

Load module: temperature field editor: **Reset to initial**

Use the following option to reset a field variable value to the value prescribed in the initial step (or to zero if no initial value was defined):

Load module: field variable editor: **Reset to initial**

Reading the Values of a Field Directly from an Alternate Input File

The data for predefined temperature, field variables, pressure stress, or mass flow rate can be contained in a separate input file (see *Input Syntax Rules*).

Input File Usage: Use one of the following options:

**TEMPERATURE*, INPUT=*file_name*
**FIELD*, INPUT=*file_name*
**PRESSURE STRESS*, INPUT=*file_name*
**MASS FLOW RATE*, INPUT=*file_name*

If the INPUT parameter is omitted, it is assumed that the data lines follow the keyword line.

Abaqus/CAE Usage: You cannot read field data from a separate input file in Abaqus/CAE.

Reading the Values of a Field from a User-Specified File

Nodal temperatures calculated during an Abaqus/Standard heat transfer or coupled thermal-electrical analysis can be used to define temperatures in a subsequent analysis. The temperatures must have been written to the results or output database file.

If nodal temperatures are written to the results file during an Abaqus/Standard heat transfer or coupled thermal-electrical analysis, they can be used to define field variables in a subsequent analysis.

In Abaqus/Standard if nodal values of temperature (NT), normalized concentrations (NNC), or electric potential (EPOT) are written to the output database file, they can be used to define field variables in a subsequent Abaqus/Standard analysis.

In Abaqus/Standard equivalent pressure stresses calculated during a mechanical analysis can be used in a subsequent mass diffusion analysis if the element output variable SINV was written to the results file averaged at the nodes (see *Element Output*).

Once the data are available in a results file or output database file, they can be read into a subsequent analysis as a predefined field. Data for field variables and pressure stress can be read from a previously generated results file. In Abaqus/Standard data can also be read from a previously generated output database file. Data for temperatures can be read from a previously generated results or output database file. Data for temperatures (and field variables in Abaqus/Standard) to be interpolated between dissimilar meshes can be read only from the output database file. The part (`.prt`) file from the original analysis is also required when reading data from the results or output database file.

When the output file of an Abaqus analysis involving beam and/or shell elements is used to define temperatures, you must ensure that the number of temperature points through the section defined for corresponding elements is consistent between the two analyses. Inconsistent temperature point definition will result in an incorrect transfer of prescribed field quantities.

Reading Field Values from a User-Specified Results File

To read field values from a user-specified results file, the data must have been written to the results file as nodal output (see *Node Output*). Only nodal quantities can be read from the results file. Since field variables can be written to the results file only as element quantities (record key 9), they cannot be read directly into a subsequent analysis. In this case you must generate a results file with the field data in the temperature record, even if the field variable in the current analysis is the same as a field variable in the previous analysis. Multiple results files must be generated for multiple field variables.

To generate the results file, you can write a program to create a results file (without running an Abaqus analysis) according to the format described in *File Output Format*. Examples of such programs are shown in that chapter. If the values will be read in as temperatures or field variables, the data must be written as nodal quantities with record key 201. If the values will be read in as a pressure stress field, the data must be averaged at the nodes (as explained in *Output to the Data and Results Files*) and written as record key 12.

Specifying the Results File to Be Read

You must specify the name of the results file from which the data are to be read for a temperature, field variable, or pressure stress. The `.fil` file extension is optional. If both `.fil` and `.odb` files exist for a temperature field and no extension is specified, the results file will be used.

Input File Usage:
**TEMPERATURE*, FILE=*file*
**FIELD*, FILE=*file*
**PRESSURE STRESS*, FILE=*file*

Abaqus/CAE Usage:
In Abaqus/CAE only predefined temperature fields and predefined field variables are available.

Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Temperature** or **Field** for the **Types for Selected Step**: select region: **Distribution: From results or output database file**, **File name:** *file*

Creating a Cyclic Temperature History

In a direct cyclic analysis in Abaqus/Standard the temperature values must be cyclic over the step: the start value must be equal to the end value. To create a cyclic temperature history from a prior heat transfer analysis that is not cyclic, you can set the starting time, f (measured relative to the total step time period, t^σ), after which the temperatures read from the results file will be ramped back to their initial condition values. At any time point $t \geq ft^\sigma$, the temperature value is equal to

$$pTemp^\theta + (1-p)\,Temp^{ini},$$

where $p = \frac{(t^\sigma - t)}{(t^\sigma - ft^\sigma)}$, $Temp^{ini}$ is the initial condition value, and $Temp^\theta$ is the interpolated value obtained from the results file at time t, as illustrated in *Figure 1*.

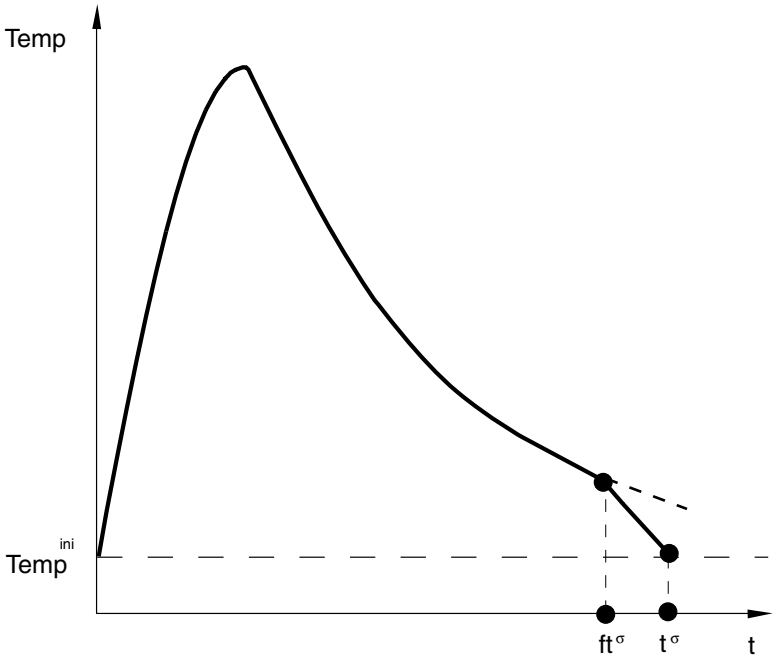


Figure 1: Ramp temperatures to their initial condition values after $t = ft^{\sigma}$ to create a cyclic temperature history.

Input File Usage: Use the following option to set the starting time for a cyclic temperature history:

**TEMPERATURE*, FILE=*file*, BTRAMP=*f*

Abaqus/CAE Usage: Cyclic temperature histories are not supported in Abaqus/CAE.

Reading Temperature Values from a User-Specified Output Database File

To read temperature values from a user-specified database file, the temperatures must have been written to the output database file (in ODB or SIM format) as nodal output (see *Writing Nodal Output to the Output Database*).

Specifying the Output Database File to Be Read for a Temperature Field

You must specify the name of the output database file (in ODB or SIM format) from which the data are to be read for a temperature field. The file extension must be included if any two of the following files exist: the results file, the ODB output database file, or the SIM database file. Only the data for the part instances that are common to both the analyses will be transferred. If the part instance names differ, you must activate the general interpolation capability.

Input File Usage: **TEMPERATURE*, FILE=*file*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**: select region: **Distribution: From results or output database file**, **File name:** *file*

Reading temperatures from a SIM database file is not supported in Abaqus/CAE.

Defining Fields Using Nodal Scalar Output Values from a User-Specified Output Database File

In Abaqus/Standard if nodal values of temperature (NT), normalized concentrations (NNC), or electric potential (EPOT) are written to the output database file, they can be used to define field variables in a subsequent Abaqus/Standard analysis. To read these values from a user-specified output database file, they must have been written to the output database file as nodal output (see *Writing Nodal Output to the Output Database*).

Specifying the Output Database File to Be Read for a Field Variable

You must specify the name of the output database file from which the data are to be read for a field variable. The `.odb` extension must be included if both results and output database files exist.

Input File Usage: **FIELD*, FILE=*file*, OUTPUT VARIABLE=*scalar nodal output variable*,

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Field** for the **Types for Selected Step**: select region: **Distribution: From results or output database file**, **File name:** *file*, **Output variable**: *scalar nodal output variable*

Interpolating Data between Meshes

Data can be mapped between the same meshes, between meshes that differ only in the element order (first-order element in heat transfer analysis and second-order element in thermal-stress analysis), or between dissimilar meshes of matching element dimensionality (solid element to solid element or shell element to shell element). If data are mapped between the same meshes, no additional computations are required. To transfer data between meshes that differ only in the element order, you must activate the midside node capability. To map data between dissimilar meshes, you must activate the general interpolation capability. The midside node capability is available only for temperatures. The midside node capability and the general interpolation capability are mutually exclusive.

Using Second-Order Stress Elements with First-Order Heat Transfer Elements (the Midside Node Capability)

In some cases it makes sense to perform an Abaqus/Standard heat transfer analysis using first-order elements followed by a thermal-stress analysis using second-order elements (and an otherwise similar mesh). For example, a heat transfer analysis including latent heat effects—for which first-order elements are best suited—can be followed by a stress analysis using second-order elements, which generally have superior deformation characteristics. In addition, the first-order temperature field calculated in the heat transfer analysis is consistent with the first-order thermal strain field provided by the second-order stress/displacement elements.

For the instances in which there is a change in the order of interpolation of element temperature variables between the heat transfer analysis and the stress analysis, temperatures must be assigned to the midside nodes of the stress/displacement elements based on the temperatures of the corner nodes of the heat transfer elements. If you specify that the midside node temperatures are needed, Abaqus will interpolate the temperatures of the midside nodes of the second-order stress/displacement elements from the corner nodes using first-order interpolation. If the midside node capability is activated in cases where both the heat transfer analysis and the stress analysis are performed with second-order elements, it is ignored. One exception is that if variable-node second-order stress/displacement elements are used in the stress analysis, activating the midside node capability will cause Abaqus to interpolate the temperatures of the midface nodes in the variable node elements from the corner or midside nodes using first-order interpolation.

Since it is assumed that the corner node temperatures have been generated in a previous heat transfer analysis, the midside node capability can be used only when the temperature field values are read from a user-specified

results or output database file. You must ensure that the nodal temperatures calculated during the heat transfer analysis are written to the results or output database file. Once the temperatures of the corner nodes are read in the subsequent stress/displacement analysis, Abaqus interpolates the midside node temperatures so that all nodes have temperatures assigned to them.

You must ensure that all temperatures of the corner nodes belonging to elements for which midside node temperatures are to be interpolated are read from the heat transfer analysis results or output database file. If the corner node temperatures are defined using a mixture of direct data input, reading from the results file or output database file, and user subroutine *UTEMP*, midside node temperatures that give unrealistic temperature fields may result. In practice, the capability for calculating midside node temperatures is most useful when temperatures generated by a heat transfer analysis are read from the results or output database file for the whole mesh during the stress analysis. Once the midside node capability is activated in a step, the capability will remain active throughout the remainder of the analysis.

Values of temperature for nodes that existed in the original analysis but do not exist in the current analysis will be ignored. Similarly, if additional nodes (but not midside nodes) exist in the current analysis, the values of fields at these nodes cannot be prescribed by reading the output files.

Input File Usage: Use the following option to interpolate temperatures between meshes that differ only in the element order:

**TEMPERATURE*, FILE=*file*, MIDSIDE

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**: select region: **Distribution: From results or output database file**, **File name:** *file*, **Mesh compatibility: Compatible**, and toggle on **Interpolate midside nodes**

Interpolating Temperatures between Dissimilar Meshes (the General Interpolation Capability)

In some cases the model for a heat transfer analysis and the model for a thermal-stress analysis may require different meshes; for example, you may want to model a smooth temperature distribution in the heat transfer analysis and stress concentration regions in the thermal-stress analysis. Both meshes have to be different and independent of each other in such cases. Abaqus offers a general interpolation capability that allows for the use of dissimilar meshes for heat transfer and thermal-stress analyses.

The interpolation is always based on the initial (undeformed) configurations. If the mesh for which the temperature field is obtained is quite different from the initial (undeformed) configuration for the thermal-stress analysis, the interpolation may not work properly even when using the tolerance parameters discussed below.

Temperatures can be interpolated between dissimilar meshes only when the temperatures are read from an output database file (in ODB or SIM format). If temperatures for nodes in the heat transfer analysis that are needed for interpolation are not written to the output database file, the values at those nodes are assumed to be zero, which may lead to incorrect results for the temperature values in the stress analysis. Similarly, if additional nodes exist in the mesh for the stress analysis, the values of temperatures at these nodes are assumed to be zero. Interpolation of temperatures can also be used for specifying temperature as a field variable in a submodel thermal-stress analysis where the temperature values are read directly from a global heat transfer analysis.

You can specify an interpolation tolerance for use in locating the nodes in the heat transfer analysis. The tolerance can be specified as an absolute value or as a fraction of the average element size. In a multistep thermal-stress analysis in which several steps read the temperature values from the same file, Abaqus interpolates the temperature values only once. If different interpolation tolerance values are used for each step, the interpolation is based on the largest specified tolerance value. If a restart analysis is performed from a particular step in the thermal-stress analysis, the restart interpolation is based on the tolerance value specified for that step.

Input File Usage: Use the following option to interpolate temperatures between dissimilar meshes:

**TEMPERATURE*, FILE=*file*`.odb`/*file*`.sim`, INTERPOLATE

Use the following option to specify the interpolation tolerance as an absolute value:

**TEMPERATURE*, FILE=*file*`.odb`/*file*`.sim`, INTERPOLATE, ABSOLUTE EXTERIOR TOLERANCE=*tolerance*

Use the following option to specify the interpolation tolerance as a fraction of the average element size:

**TEMPERATURE*, FILE=*file*`.odb`/*file*`.sim`, INTERPOLATE, EXTERIOR TOLERANCE=*tolerance*

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**: select region: **Distribution: From results or output database file**, **File name:** *file*`.odb`, **Mesh compatibility: Incompatible**, exterior tolerance: **absolute** or **relative** *tolerance*

Interpolating temperatures from a SIM database file is not supported in Abaqus/CAE.

Interpolating Temperatures between Dissimilar Meshes with User-Specified Regions

When regions of elements in the heat transfer analysis are close or touching, the dissimilar mesh interpolation capability can result in an ambiguous temperature association. For example, consider a node in the current model that lies on or close to a boundary between two adjacent parts in the heat transfer model, and consider a case where temperatures in these parts are different. When interpolating, Abaqus will identify a corresponding parent element at the boundary for this node from the heat transfer analysis. This parent element identification is done using a tolerance-based search method. Hence, in this example the parent element might be found in either of the adjacent parts, resulting in an ambiguous temperature definition at the node. You can eliminate this ambiguity by specifying the source regions from which temperatures are to be interpolated. The source region refers to the heat transfer analysis and is specified by an element set. The target region refers to the current analysis and is specified by a node set.

Input File Usage: Use the following option to interpolate temperatures between dissimilar meshes with user-specified regions:

**TEMPERATURE*, FILE=*file*`.odb`/*file*`.sim`, INTERPOLATE, DRIVING ELSETS

Abaqus/CAE Usage: You cannot specify the regions where temperatures are to be interpolated in Abaqus/CAE.

Interpolating Scalar Nodal Output Variables between Dissimilar Meshes (the General Interpolation Capability) onto Field Variables in Abaqus/Standard

Abaqus/Standard offers a general interpolation capability that allows for nodal values of temperature, normalized concentration, and electric potential from one analysis to be mapped onto field variables in a subsequent analysis in the cases where the meshes in the two analyses are dissimilar.

The interpolation is always based on the initial (undeformed) configurations. If the mesh for which the field variable is obtained is quite different from the initial (undeformed) configuration for the original analysis, the interpolation may not work properly even when using the tolerance parameters discussed below.

Temperatures, normalized concentrations, and electric potentials can be interpolated between dissimilar meshes onto field variables only when they are read from an output database file. If scalar values for nodes in the current analysis that are needed for interpolation are not written to the output database file, the values at those nodes are assumed to be zero, which may lead to incorrect results for the field variables. Similarly, if additional nodes exist in the mesh for the current analysis, the values of the field variables at these nodes are assumed to be zero.

You can specify an interpolation tolerance for use in locating the nodes in the original analysis. The tolerance can be specified as an absolute value or as a fraction of the average element size. In a multistep analysis in which several steps read nodal output variables values from the same file, Abaqus interpolates the nodal values only once. If different interpolation tolerance values are used for each step, the interpolation is based on the largest specified tolerance value. If a restart analysis is performed from a particular step in the original analysis, the restart interpolation is based on the tolerance value specified for that step.

Input File Usage: Use the following option to interpolate scalar nodal output variables between dissimilar meshes:

**FIELD*, FILE=*file*`.odb`, OUTPUT VARIABLE=*scalar nodal output variable*, INTERPOLATE

Use the following option to specify the interpolation tolerance as an absolute value:

**FIELD*, FILE=*file*`.odb`, OUTPUT VARIABLE=*scalar nodal output variable*, INTERPOLATE, ABSOLUTE EXTERIOR TOLERANCE=*tolerance*

Use the following option to specify the interpolation tolerance as a fraction of the average element size:

**FIELD*, FILE=*file*`.odb`, OUTPUT VARIABLE=*scalar nodal output variable*, INTERPOLATE, EXTERIOR TOLERANCE=*tolerance*

Abaqus/CAE Usage: Specifying an interpolation tolerance for predefined field variables is not supported in Abaqus/CAE.

Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Field** for the **Types for Selected Step**: select region: **Distribution: From results or output database file**, **File name:** *file*`.odb`, **Output variable**: *scalar nodal output variable*, **Mesh compatibility: Incompatible**

Specifying the Step and Increment to Be Read from the File

You can specify the first and last step, respectively, from which results will be read. Similarly, you can specify the first and last increment, respectively, from which results will be read. You can specify any combination of these values. Any zero-increment file output that is present in the results file of an Abaqus/Standard analysis (written only if the zero increment results are requested; see *Obtaining Results at the Beginning of a Step*) will be ignored. Results must have been written to the results or output database file at the specified step and increment.

If you do not specify the first step from which to read, Abaqus will begin reading results from the first step available in the results or output database file.

If you do not specify the first increment from which to read, Abaqus will begin reading results from the first increment available in the first step from which results will be read (the first increment following the zero increment if zero-increment file output is present in the results file).

If you do not specify the last step from which to read, the first step from which results will be read will also be the last step.

If you do not specify the last increment from which to read, Abaqus will read the results or output database file until it reaches the last available increment in the last step from which results will be read.

Input File Usage: Use one of the following options:

**TEMPERATURE*, FILE=*file*, BSTEP=*bstep*, BINC=*binc*, ESTEP=*estep*, EINC=*einc*
**FIELD*, FILE=*file*, BSTEP=*bstep*, BINC=*binc*, ESTEP=*estep*, EINC=*einc*
**PRESSURE STRESS*, FILE=*file*, BSTEP=*bstep*, BINC=*binc*, ESTEP=*estep*, EINC=*einc*

For example, the following input would read temperature data from output database file `heat.odb` beginning at Step 2, increment 2, and ending at Step 3, increment 5:

**TEMPERATURE*, FILE=`heat.odb`, BSTEP=2, BINC=2, ESTEP=3, EINC=5

Abaqus/CAE Usage: In Abaqus/CAE only predefined temperature fields and predefined field variables are available.

Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Temperature** or **Field** for the **Types for Selected Step**: select region: **Distribution: From results or output database file**, **File name:** *file*, **Begin step:** *bstep*, **Begin increment:** *binc*, **End step:** *estep*, and **End increment:** *einc*

Interpolation in Time

When Abaqus reads temperature, field variable, or equivalent pressure stress data from a results file or temperatures from an output database file, it must obtain values of the field at the time points used by the analysis. Since data corresponding to these time points are usually not present in the results or output database files, Abaqus will interpolate linearly in time between the time points stored in the file to obtain values at the time points required by the analysis. Since the interpolation is linear, you must take care to provide sufficient data in the results or output database file to make this interpolation meaningful.

For the purpose of such interpolation the time period of the results being read in is determined as follows:

- The period starts at the time of the most recent increment written, of the relevant field, that precedes the beginning increment (either user-specified or default). For example if your results file contains temperature field data at increments 5, 10, and 15; and you specify a beginning increment number of 10 when reading these results; the results period starts with the time associated with increment 5 since that is the most recent increment that precedes the specified beginning increment of 10. You can ensure that the results starting time matches the beginning time of the beginning increment you specify by writing the results data with an increment frequency of 1.
- The period ends at the completion of the ending increment (either user-specified or default).

If the analysis requires data at a time point prior to the first increment for which data are available in the either of files, Abaqus will interpolate between the given initial condition data and the data of the first increment stored in the file.

Reading Results for Multiple Fields

If data for multiple fields are being read in the same step and the time values corresponding to the starting step and increment or to the ending step and increment are different for different fields, Abaqus interpolates through the total time period from the earliest time point chosen in any file to the latest. For example, suppose the starting

increment in the starting step in the temperature file begins at 3 sec and the ending increment in the ending step ends at 6 sec. During the same step we also read field variable data, for which the starting increment in the starting step begins at 2 sec and the ending increment in the ending step ends at 5 sec. In such a case the time period used for interpolation is from 2 sec to 6 sec.

Automatic Adjustment of the Time Scale

It is convenient to set the period of the step equal to the time period of the files being read in. Otherwise, Abaqus will automatically scale the time period from the results or output database file to match the time period of the stress analysis. The scale factor is t^{σ}/t^{θ}, where t^{σ} is the time period of the stress analysis and t^{θ} is the total time period obtained from all results or output database files, as described above.

Obtaining Results at a Particular Point in Time

In Abaqus/Standard it is sometimes desirable to carry out a calculation corresponding to the field values at a particular point in time. For example, suppose that temperature data are available in the output file for increment 10 at time $t = 4$ and increment 15 at time $t = 5$ and that you wish to carry out a static analysis based on temperature values at $t = 4.5$. In this case Abaqus must interpolate linearly between the results at $t = 4$ and $t = 5$ to obtain the intermediate result at $t = 4.5$. To accomplish this task, you should specify an initial time increment of 4.5 and a time period of 5. for the static analysis step and read the temperature values from the output file starting at Step 1, Increment 1 and ending at Step 1, Increment 15. Specifying a starting increment of 1 instead of 10 ensures that t^{θ} is the entire time period stored in the output file, not just the period between increments 10 and 15; hence, the scale factor between the output file data and the static analysis is unity, and the initial time of 4.5 has the desired meaning.

Initial Transients

To track initial transients accurately, Abaqus/Standard may automatically reduce the initial time increment for the step. If the user-specified suggested initial time increment is greater than the scaled value of the first time increment read from the Abaqus/Standard results file, Abaqus/Standard will use that scaled value.

Restrictions

The following restrictions exist:

- Temperatures and field variables cannot be read from a user-specified file in a modified Riks static analysis step (*Unstable Collapse and Postbuckling Analysis*).
- Temperature cannot be interpolated from a coupled thermal-electrical analysis.
- Equivalent pressure stress cannot be read from the results file if the model is defined in terms of an assembly of part instances.
- In Abaqus/Explicit field variables cannot be read from the output database file.
- Pressure stress cannot be read from the output database file.
- Elements that do not support interpolation for temperature mapping include the complete libraries of convective heat transfer elements, axisymmetric elements with nonlinear axisymmetric deformation, axisymmetric surface elements, truss elements, beam elements, link elements, hydrostatic fluid elements, solid infinite stress elements, and coupled thermal/electrical elements. Other specific elements that are not supported include: GKPS6, GKPE6, GKAX6, GK3D18, GK3D12M, GK3D4L,GK3D6L, GKPS4N, GKAX6N, GK3D18N, GK3D12MN, GK3D4LN, and GK3D6LN.

Defining the Values of a Predefined Field in a User Subroutine

In Abaqus/Standard you can specify predefined temperatures, field variables, equivalent pressure stresses, or mass flow rates at the nodes in a user subroutine. Temperature values can be defined in user subroutine *UTEMP*; field variable values, in user subroutine *UFIELD*; equivalent pressure stress values, in user subroutine *UPRESS*; and mass flow rates, in user subroutine *UMASFL*.

In Abaqus/Explicit you can specify predefined field variables in user subroutine *VUFIELD*.

The user subroutine (*UTEMP*, *UFIELD*, *VUFIELD*, *UPRESS*, or *UMASFL*) will be called for each specified node. Field values entered directly will be ignored. If a results or output database file has been specified in addition to the user subroutine, values read from the results or output database file will be passed into the user subroutine for possible modification.

Input File Usage: Use one of the following options:

**TEMPERATURE*, USER
**FIELD*, USER
**PRESSURE STRESS*, USER
**MASS FLOW RATE*, USER

Abaqus/CAE Usage: In Abaqus/CAE only predefined temperature fields and predefined field variables are available.

Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Temperature** or **Field** for the **Types for Selected Step**: select region: **Distribution: User-defined** or **From results or output database file and user-defined**

Updating Multiple Predefined Field Variables

If multiple field variables are predefined, only one field variable at a time can be redefined in user subroutine *UFIELD* or *VUFIELD*. There are situations in which the analysis requires a number of field variables that are predefined with respect to the solution but depend on each other. You can specify the number of field variables to be updated simultaneously at a point, *n*. Abaqus passes information about *n* field variables at each specified node into *UFIELD* or *VUFIELD*.

You can update all or part of the field variables used in the analysis but must remember that the field variables are numbered consecutively from 1. If, for example, you have four field variables in the analysis and want to update the second and third variables simultaneously in user subroutine *UFIELD*, you must specify *n*=3. In this case Abaqus/Standard passes information about the first three field variables into user subroutine *UFIELD*, and you update only the second and third variables.

Input File Usage: **FIELD*, USER, NUMBER=*n*

Abaqus/CAE Usage: Allowing multiple predefined field variables to be updated simultaneously in a user subroutine is not supported in Abaqus/CAE.

Defining Solution-Dependent Field Variables

In Abaqus/Standard solution-dependent field variables can be defined in user subroutine *USDFLD*. The values of predefined field variables or initial fields can be passed into user subroutine *USDFLD* and can be changed in that routine—see *Material Data Definition*.

Changes to the field variables in *USDFLD* are local to the material point and do not affect the nodal values.

Data Hierarchy

If both results or output database file input and direct data input are used in the same step, the direct data input will take precedence if both define the field at the same node. If user subroutine input is specified, the values given directly are ignored and the user subroutine modifies the values read from the results or output database file.

Element Type Considerations

You can specify either one or several values of a predefined field at a node, depending on the element type that is used. You should note the following considerations when choosing the form of predefined field specification.

Use in a Mass Diffusion Analysis

For solid elements only one value can be given at a node. Since only solid elements can be used in mass diffusion analysis, this is the only way to define equivalent pressure stresses at a node.

Use with Beam and Shell Elements

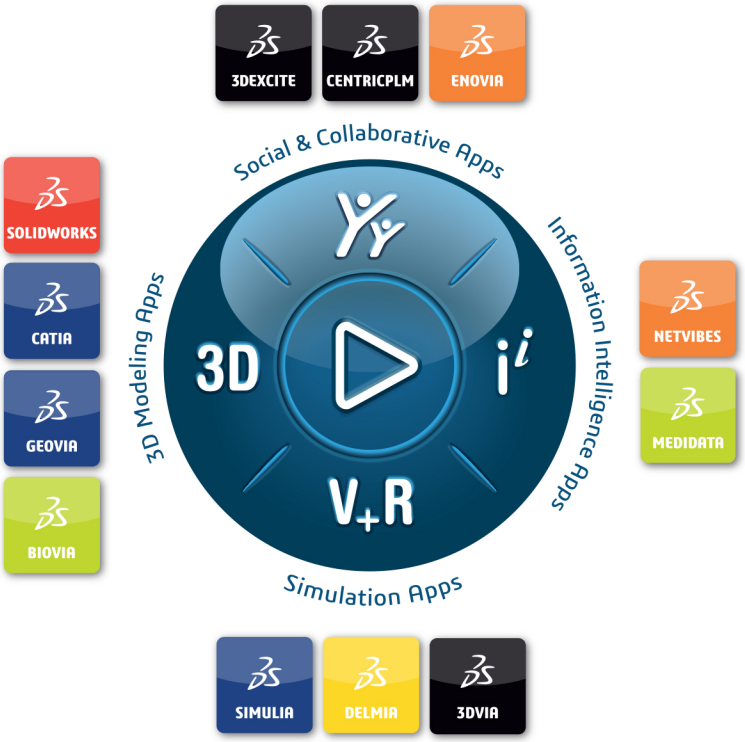
The following possibilities exist for temperatures and field variable specification in beam and shell elements:

- For shell and beam elements with general cross-section definitions, the temperature and field variable magnitude at points in the section is defined by the value at the reference surface. Any gradient of these variables specified across the section is ignored.
- For shell and beam elements with cross-sections that require numerical integration, the temperature and field variable magnitudes at points in the section can be defined either from the value at the reference surface and the gradient or gradients across the section or by giving the values at a number of points across the section. The choice between these two methods is made in the section definition (see *Specifying Temperature and Field Variables* and *Specifying Temperature and Field Variables* for details).

See *About the Element Library* for the details of use with each element type. The default, if only one value is given, is a constant magnitude across the section.

Temperature and Field Variable Compatibility across Elements

Abaqus assumes that the field definitions (including initial conditions) at all the nodes of any element are compatible with the field definition method chosen for the element. Cases may arise where the definition of a field changes from one element to the next (for example, when two adjacent shell elements have a different number of section points through the thickness or when the temperature and field variable magnitudes for one beam element are defined by giving the values at a number of points across the section while those for the abutting beam element are defined from the value at the reference surface and the gradient or gradients across the section). In these cases separate nodes should be used on the interface between such elements and multi-point constraints should be applied to make the displacements and rotations the same at corresponding nodes (see *General Multi-Point Constraints*); otherwise, the fields on the nodes at the interface will be used for each adjacent element with the field definition method chosen for the element.



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.