SIMULIA

3DEXPERIENCE

ABAQUS 2026 FD04
INTERACTIONS GUIDE

DASSAULT SYSTEMES

Contents

Trademarks and Legal Notices....5
Abaqus Interactions Guide....6
What's New....7
Defining Contact Interactions....25
About Contact Interactions....26
Available Contact Algorithms in Abaqus....26
Defining a Surface-Based Contact Simulation....26
Contact-Dependent Loading....28
Contact Simulation Capabilities in Abaqus/Standard....28
Contact Simulation Capabilities in Abaqus/Explicit....31
Compatibility between Abaqus/Standard and Abaqus/Explicit....33
Defining General Contact in Abaqus/Standard....35
About General Contact in Abaqus/Standard....36
Surface Properties for General Contact in Abaqus/Standard....66
Contact Properties for General Contact in Abaqus/Standard....76
Stabilization for General Contact in Abaqus/Standard....79
Defining Contact Pairs in Abaqus/Standard....84
About Contact Pairs in Abaqus/Standard....85
Assigning Surface Properties for Contact Pairs in Abaqus/Standard....100
Assigning Contact Properties for Contact Pairs in Abaqus/Standard....102
Modeling Contact Interference Fits in Abaqus/Standard....104
Defining Tied Contact in Abaqus/Standard....110
Extending Main Surfaces and Slide Lines....113
Contact Modeling If Substructures Are Present....116
Contact Modeling If Asymmetric-Axisymmetric Elements Are Present....119
Defining General Contact in Abaqus/Explicit....120
About General Contact in Abaqus/Explicit....121
Assigning Surface Properties for General Contact in Abaqus/Explicit....136
Assigning Contact Properties for General Contact in Abaqus/Explicit....159
Defining Contact Pairs in Abaqus/Explicit....162
About Contact Pairs in Abaqus/Explicit....163
Assigning Surface Properties for Contact Pairs in Abaqus/Explicit....171
Assigning Contact Properties for Contact Pairs in Abaqus/Explicit....178
Contact Property Models....180
Mechanical Contact Properties....181
About Mechanical Contact Properties....182
Contact Pressure-Overclosure Relationships....186
Contact Damping....194
Contact Blockage....199
Frictional Behavior....201
User-Defined Interfacial Constitutive Behavior....226
Interaction of Debonded Surfaces....232
Contact Cohesive Behavior....234

Contact Wear....259
Thermal Contact Properties....267
Thermal Properties in a Contact Property Definition....267
Contact Conductance between Surfaces....268
Thermal Contact Considerations in Abaqus/Explicit....272
Modeling Radiation between Surfaces When the Gap Is Small....273
Modeling Heat Generated by Nonthermal Surface Interactions....275
Surface-Based Interaction Variables for Thermal Contact Property Models....276
Thermal Interaction Variables for Thermal Gap Elements....278
Thermal Interactions Involving Rigid Bodies....278
Modeling Thermal Interactions with Node-Based Surfaces....278
Thermal Interactions between Surfaces with Nodes Containing Multiple Temperature Degrees of Freedom.278
Electrical Contact Properties....279
Electrical Properties in a Contact Property Definition....279
Electrical Conductance between Surfaces....280
Modeling Heat Generated by Electrical Conduction between Surfaces....283
Including Ion or Species Concentration Properties in a Contact Property Definition....283
Controlling the Distance within Which Ionic/Species Flux Contact Properties Are Active....284
Controlling Gap Diffusivity Associated with Ionic/Species Flux across a Contact Interface....285
Defining Gap Diffusivity as a Function of Predefined Field Variables....286
Surface-Based Output Variables for Electrical Contact Property Models....286
Pore Fluid Contact Properties....288
Contact Pressure in Pore Fluid Interactions....288
Including Pore Fluid Properties in a Contact Property Definition....289
Controlling the Distance within Which Pore Fluid Contact Properties Are Active....289
Controlling Contact Permeability Associated with Fluid Flow across a Contact Interface....290
Controlling the Constraint Enforcement Method....291
Coupled Heat Transfer–Pore Fluid Contact Properties....292
Output....292
Contact Initialization....293
Contact Initialization for General Contact in Abaqus/Standard....294
Default Contact Initialization Method....294
Defining Alternative Contact Initialization Methods....295
Assigning Contact Initialization Methods....301
Contact Initialization for Contact Pairs in Abaqus/Standard....304
Treatment of Initial Overclosures and Unintended Initial Gaps....304
Adjusting Initial Surface Positions to Resolve Small Initial Gaps or Overclosures....305
Storing Penetration Offsets for Noncompliant Surfaces to Be Initially in Contact....307
Initial Clearance or Overclosure for Small-Sliding Contact....308
Alternative Methods for Specifying Precise Initial Clearances or Overclosures....308
Additional Contact Initialization Options for Small-Sliding Contact in Abaqus/Standard....310
Defining Precise Initial Clearance or Overclosure Values....310
Specifying a Uniform Clearance or Overclosure for the Surfaces....310
Specifying Spatially Varying Clearances or Overclosures for the Surfaces....311
Generating Contact Normal Directions Based on a Reference Thread Geometry....313
Visualizing the Precise Initial Clearances or Overclosures....316
Contact Initialization for General Contact in Abaqus/Explicit....317
Defining and Assigning Contact Initialization Methods....317

Controlling Initial Gap or Penetration State....318
Resolving Discrepancies between Mesh Geometry and Desired Initial Clearance or Penetration State.322
Contact Initialization Search Criteria....324
Contact Initialization Details Specific to Shells....326
Alternative Contact Initialization User Interface....328
Contact Initialization for Contact Pairs in Abaqus/Explicit....330
Adjustments of Overclosed Surfaces in the First Step of the Simulation....330
Adjustments of Overclosed Surfaces during Subsequent Steps in the Simulation....332
Specifying Initial Clearance Values Precisely....332
Contact Formulations and Numerical Methods....335
Contact Formulations, Numerical Methods, and Controls in Abaqus/Standard....336
Contact Formulations in Abaqus/Standard....337
Contact Constraint Enforcement Methods in Abaqus/Standard....357
Smoothing Contact Surfaces in Abaqus/Standard....369
Initial Contact Stresses in Abaqus/Standard....377
Generally Applicable Contact Controls in Abaqus/Standard....379
Contact Controls Specific to General Contact in Abaqus/Standard....388
Contact Formulations, Numerical Methods, and Controls in Abaqus/Explicit....393
Contact Formulation for General Contact in Abaqus/Explicit....394
Contact Formulations for Contact Pairs in Abaqus/Explicit....400
Contact Constraint Enforcement Methods in Abaqus/Explicit....412
Contact Controls for General Contact in Abaqus/Explicit....419
Contact Controls for Contact Pairs in Abaqus/Explicit....426
Contact Difficulties and Diagnostics....430
Resolving Contact Difficulties in Abaqus/Standard....431
Contact Diagnostics in an Abaqus/Standard Analysis....432
Common Difficulties Associated with Contact Modeling in Abaqus/Standard....442
Resolving Contact Difficulties in Abaqus/Explicit....456
Contact Diagnostics in an Abaqus/Explicit Analysis....457
Common Difficulties Associated with Contact Modeling Using Contact Pairs in Abaqus/Explicit....461
Contact Elements in Abaqus/Standard....469
Contact Modeling with Elements....470
Gap Contact Elements....471
Gap Contact Elements....472
Gap Element Library....479
Tube-to-Tube Contact Elements....482
Tube-to-Tube Contact Elements....483
Tube-to-Tube Contact Element Library....486
Slide Line Contact Elements....490
Slide Line Contact Elements....491
Axisymmetric Slide Line Element Library....496
Rigid Surface Contact Elements....499
Rigid Surface Contact Elements....500
Axisymmetric Rigid Surface Contact Element Library....503
Cavity Radiation in Abaqus/Standard....505
Defining a Cavity Radiation Problem....506
Model Definition....506

History Definition....506
Defining Surfaces....507
Restrictions....507
Controlling Spurious Spatial Oscillations....507
Defining Surface Radiation Properties....508
Defining the Emissivity....508
Defining the Stefan-Boltzmann Constant and Value of Absolute Zero....509
Constructing a Cavity....509
Setting Surface Properties....509
Creating a Closed Cavity....509
Creating an Open Cavity....510
Decomposing Large Cavities in Parallel....510
Solving Radiative Heat Transfer Equations in Parallel....511
Convergence of Models with Decomposed Cavities....511
Kinematic Constraints on Models with Decomposed Cavities....511
Defining Cavity Symmetries....511
Reflection Symmetry....512
Periodic Symmetry....514
Cyclic Symmetry....516
Combining Symmetries....518
Prescribing Motion during a Cavity Radiation Analysis....520
Translational Motion....521
Rotational Motion....521
Simultaneous Translational and Rotational Motion....522
Controlling the Time Variation of the Motion....524
Controlling the Frequency of View Factor Recalculation due to Motion....524
Controlling View Factor Calculation during the Analysis....525
Activating and Deactivating Cavity Radiation....525
Controlling the Accuracy of View Factor Calculations....526
Controlling Checks for Surface Blocking....527
Reducing Computations for Surfaces That Are Far Apart....528
Memory Usage in Cavity Radiation Analyses....528
Initial Conditions....529
Boundary Conditions....529
Loads....529
Predefined Fields....529
Material Options....530
Elements....530
Output....530
Writing the View Factor Matrices to the Results File....531
Requesting Surface Variable Output....531
Input File Template....532

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 Interactions Guide

The Abaqus Interactions Guide discusses the contact and interaction models available in Abaqus.

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.

Abaqus provides more than one approach for defining contact. Abaqus/Standard includes the following approaches for defining contact:

- general contact;
- contact pairs; and
- contact elements.

Abaqus/Explicit includes the following approaches for defining contact:

- general contact; and
- contact pairs.

Each approach has somewhat unique advantages and limitations.

What's New

This page describes recent changes in Abaqus Interactions.

2026 FD04

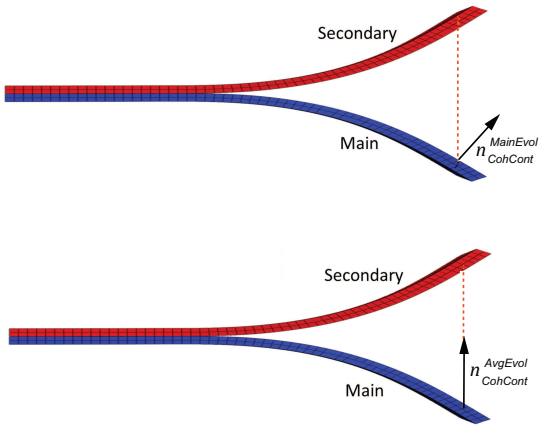
Cohesive Contact Calculations Involving Large Cohesive Strains

When modeling ductile cohesive contact bonds in Abaqus/Explicit, you can now specify that the main and secondary surfaces both influence the evolution of the cohesive contact normal direction equally.

Benefits: Modeling of ductile cohesive contact bonds in Abaqus/Explicit provides results more similar to cohesive element modeling.

In earlier releases, the contact normal direction for Abaqus/Explicit contact with cohesive behavior was always based solely on the main surface. This behavior remains the default. However, you can now specify that the cohesive contact normal direction be based on equal influence of the evolution normal directions of both the main and secondary surfaces.

The figure below shows deformed configurations for peeling test simulations with ductile cohesive contact bonds. The two parts are flat initially (not shown) with cohesive contact along the entire length of the interface. Symmetry exists in the mesh refinement and loading conditions. In the top image, the cohesive contact normal direction evolves with only the main surface such that the cohesive strain develops normal and shear components. In the bottom image, the cohesive strain is aligned with the cohesive contact normal direction, which evolves taking into consideration changes in both the main and secondary surface normal directions, resulting in cohesive strain with no shear.



For more information, see **CONTACT CONTROLS ASSIGNMENT*, *Contact Cohesive Behavior*, and *Cohesive surface interaction*.

Modeling Threaded Bolt Connections with Surface Element Representations of Thread Geometry

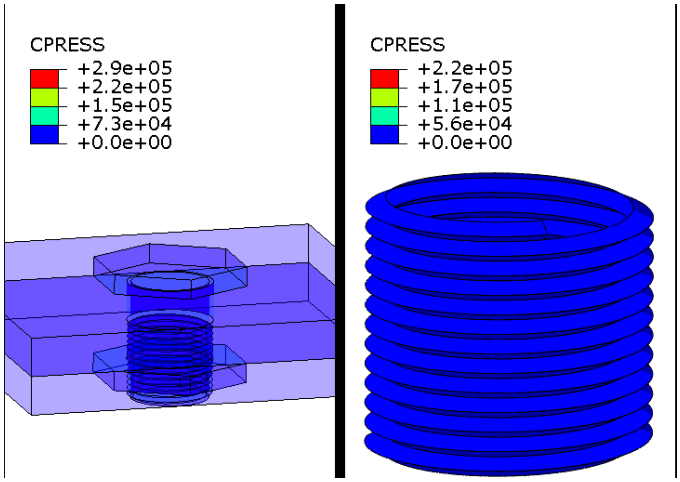
You can now specify an enhanced approach for modeling bolted connections with general contact in Abaqus/Standard that allows thread geometry to be created automatically using surface elements on top of nominal meshes.

Benefits: You can account for the true thread geometry for accurate and efficient modeling of large relative motions between the mating internal and external threads in bolt tightening and loosening simulations.

Refined solid meshes to capture thread geometry accurately on bolts and nuts in a bolted connection can make bolted connection simulations expensive. For efficiency, Abaqus allows you to specify nominal meshes with cylindrical surfaces to represent the internal and external mating surfaces for use with an approximate method to account for thread geometry. However, when large relative motion occurs between the threads during bolt tightening and loosening, the approximate method based on specifying bolt clearances with a small-sliding tracking approach might be inaccurate. The new approach generates thread meshes with surface elements that are constrained automatically to the user-defined nominal cylindrical meshes.

The generated thread meshes inherit the contact properties and initialization definitions from the nominal meshes. By default, finite-sliding contact is activated on the generated thread surfaces to model large relative motion accurately while maintaining efficiency.

The follow animation shows a bolt tightening simulation where the thread meshes are generated automatically for both the bolt and the nut based on the description of the thread geometry parameters:



For more information, see *About General Contact in Abaqus/Standard*.

2026 FD02

Mapping of Cohesive Contact State from Abaqus/Standard to Abaqus/Explicit Upon Import

You can now import the cohesive contact status and damage state from Abaqus/Standard to Abaqus/Explicit.

Benefits: You can more accurately model bonding behavior using workflows that involve import from Abaqus/Standard to Abaqus/Explicit.

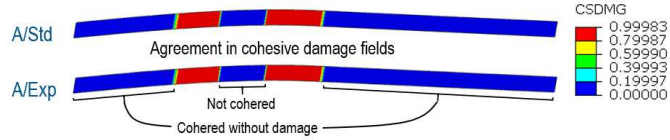
Cohesive contact status and damage are now mapped across import from Abaqus/Standard to Abaqus/Explicit, despite various differences in individual contact constraints between these two solvers. In earlier releases, the cohesive contact state (CSTATUS) was independently initialized in Abaqus/Explicit upon import from

Abaqus/Standard, such that CSTATUS would often switch to a "Not in contact" state where a tensile cohesive contact state existed prior to import.

The figure shows contour plots of CSTATUS of a partially bonded interface for an Abaqus/Standard simulation at the point of import (top) and the mapped state of the cohesive contact status upon import to Abaqus/Explicit (bottom).

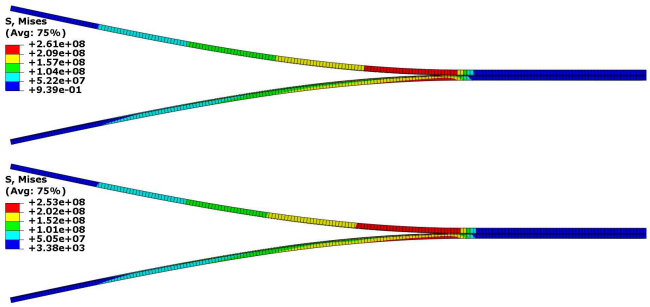


The figure below shows contour plots of cohesive contact damage (CSDMG) of a partially bonded interface for an Abaqus/Standard simulation at the point of import (top) and the mapped state of cohesive contact damage upon import to Abaqus/Explicit (bottom).



As expected, this enhancement results in consistent contour plots of contact status (CSTATUS) and cohesive damage (CSDMG) across Abaqus/Standard and Abaqus/Explicit import.

The figure below shows a comparison of Mises stress and overall delamination for the Abaqus/Standard simulation (top) and the imported Abaqus/Explicit simulation (bottom) after post-import loading.



For more information, see *Transferring Results between Abaqus/Explicit and Abaqus/Standard* and *Delamination analysis of laminated composites*.

Enhanced Contact Mass Scaling in Abaqus/Explicit

The nodal mass added due to contact mass scaling is now permanent.

Benefits: The new algorithm eliminates spurious oscillations and sudden instabilities in models with rapidly changing contact status.

Contact mass scaling is now applied permanently rather than incrementally during the analysis, which greatly improves stability and reduces artificial oscillations in models sensitive to contact chattering. The small permanent mass increase is generally negligible (and very localized) except for cases where the contact penalty stiffness is scaled up significantly.

For more information, see *Mass Scaling to Account for Contact Stiffness*.

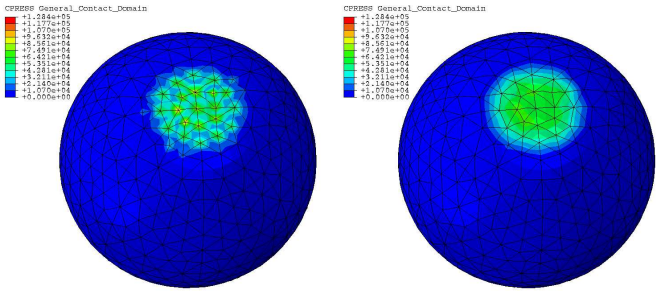
2026 FD01

Enhanced Contact Stress for C3D10 Elements in Abaqus/Explicit

Contact stresses are now smoothed during postprocessing operations for second-order tetrahedral elements (C3D10) in Abaqus/Explicit.

Benefits: The contact stress representation is more continuous and visually coherent, and it is consistent with the visualization in Abaqus/Standard.
Contact stress distribution is a key factor in many applications, including load transfer, wear, material failure, and interface behavior in mechanical analyses.

The example below illustrates a sphere compressed between two analytical rigid surfaces (not shown). The contact distributions clearly demonstrate the benefits of the new filtered output.



For more information, see *General Contact Output Variables*.

Separate Numerics and Physics in Contact Keyword Interface

Specific numerical controls of contact-related capabilities in the contact keyword interface have become separate from keywords that control the physics related to contact.

Benefits: The new contact keyword interface allows for numerical controls to be defined in more localized regions of the model.

The new contact keyword interface separates specific numerical contact controls from the physics controls. Physics related to contact continues being defined through interaction properties, while numerical controls are defined surface pairwise with contact controls assignment. Specifically, there are four numerical controls that have a new contact keyword interface: the contact enforcement method in Abaqus/Standard, the method of friction enforcement of the “stick” condition in Abaqus/Standard, the contact permeability constraint method in Abaqus/Standard, and the tracking thickness used to extend the range for which contact opening output is provided.

Th contact enforcement in Abaqus/Standard can be defined as direct, augmented Lagrange, linear penalty, or nonlinear penalty. The new surface pairwise method using contact controls assignment is mutually exclusive with the interaction property method of defining enforcement. For example, an error message occurs if user specified contact enforcement defined is using both the new interface and the previous interface for the same surface pairings in the contact domain.

The friction enforcement of the “stick” condition in Abaqus/Standard can be defined as penalty with a unitless slip tolerance, penalty with a slip tolerance distance, and Lagrange. The new surface pairwise method using contact controls assignment is mutually exclusive with the interaction property method of defining friction enforcement of the “stick” condition. For example, an error message occurs if user specified friction enforcement defined is using both the new interface and the previous interface for the same surface pairings in the contact domain.

The contact permeability constraint method in Abaqus/Standard is defined as direct or penalty. The new surface pairwise method using contact controls assignment is mutually exclusive with the interaction property method of defining the contact permeability constraint method. For example, an error message occurs if user specified contact permeability constraint method defined is using both the new interface and the previous interface for the same surface pairings in the contact domain.

The tracking thickness assigns a thickness that determines the contact tracking distance between surfaces. The new surface pairwise method using contact controls assignment is mutually exclusive with the interaction property method of defining the tracking thickness. For example, an error message occurs if the user specified tracking thickness defined is using both the new interface and the previous interface for the same surface pairings in the contact domain.

For more information, see **CONTACT CONTROLS ASSIGNMENT*, *Contact Constraint Enforcement Methods in Abaqus/Standard*, *Frictional Behavior*, *Pore Fluid Contact Properties*, and *About General Contact in Abaqus/Explicit*.

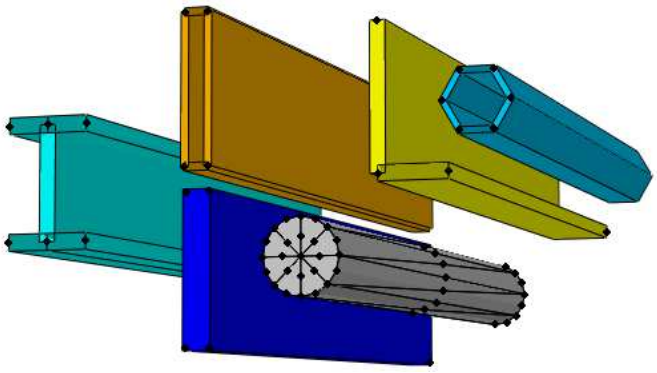
2026 GA

Contact Surface Representation for Beams with Circular Cross-Sections in Abaqus/Standard

You can configure Abaqus/Standard to internally generate surface elements that represent the circular cross-section of beams and pipes for use in general contact calculations.

Benefits: You can achieve more accurate modeling of beam ends and obtain detailed contact output on the contact surface.

In earlier releases, Abaqus/Standard automatically generated a mesh of contact faces, nodes, and edges near the exterior of noncircular beam cross-sections by default. Now, you can configure Abaqus/Standard to internally generate surface elements that represent the circular cross-sections of beams and pipes for general contact calculations. The image below shows contact surface representations for beams with various cross-sections based on surface element meshes that are internally generated by Abaqus/Standard for general contact.



For more information, see *About General Contact in Abaqus/Standard* and **CONTACT CONTROLS ASSIGNMENT*.

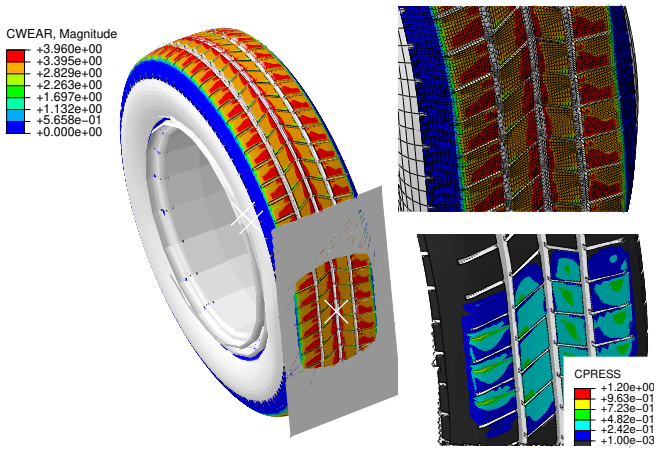
2025 FD03

Contact Wear with Steady-State Transport in Abaqus/Standard

You can now specify surface wear properties directly in steady-state transport analyses, and Abaqus/Standard calculates the wear rate. You can request output of the resulting total wear distances.

Benefits: It is much simpler to simulate wear accumulation in steady-state transport analyses with wear coefficients specified as surface properties and wear calculations done by Abaqus, compared to a preexisting approach involving user subroutine *UMESHMOTION*.

The animation below shows a passenger car tire rolling at 70 km/h and accumulating wear over a distance of 58,880 km. The 3D tire model is generated from periodic symmetric-model generation of a wedge with repeating tread configurations. Surface wear properties are specified on the tread surface. The wear distance contours show that a maximum wear distance of 3.96 mm is accumulated, which is approximately 63% of the tread depth. The animation also shows that the contact footprint expands as wear accumulates and the wear rate slows down as the contact foot print expands due to redistribution of contact stresses. In the wear calculations shown here, wear distances are treated as offsets for contact penetration calculations and the nodal locations of the underlying elements due to wear are not modified. With this built-in capability, Abaqus manages complex details involving streamline calculations that would be very challenging to manage within a user subroutine.



For more information, see *Contact Wear*, *Contact wear modeling*, and *Tread wear simulation in Abaqus/Standard*.

2025 FD01

Contact Involving Beams with Noncircular Cross-Sections in Abaqus/Standard

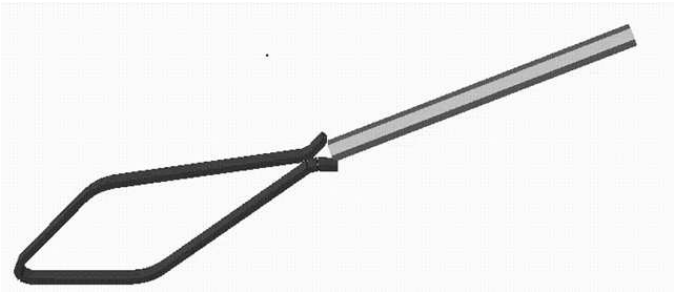
Contact calculations now consider realistic beam cross-section geometry in Abaqus/Standard by default. This change makes the behavior in Abaqus/Standard consistent with Abaqus/Explicit.

Benefits: The new default contact treatment of beams with noncircular cross-sections is more realistic.

In earlier releases, Abaqus/Standard approximately represented beams in contact calculations with nodes and edges of the beam mesh having a radius encompassing the actual beam cross-section.

The new default beam representation for contact uses an automatically generated mesh of contact faces, nodes, and edges close to the exterior of noncircular beam cross-sections. Some surface thickness is assigned automatically, which provides an accurate contact representation of the beam geometry with some rounding of edges.

The animation below shows a hexagonal cross-section beam inserted into the end of a part resembling tweezers and then rotated. Both parts are modeled with beam elements. In this example, the beam elements of the tweezers have a rectangular cross-section. The response of the tweezers to the insertion and rotation of the other part is intuitive. Realistic representation of the beam contact surfaces is crucial to the accuracy of this simulation.



You can revert to the previous default behavior, as discussed in **CONTACT CONTROLS ASSIGNMENT*.

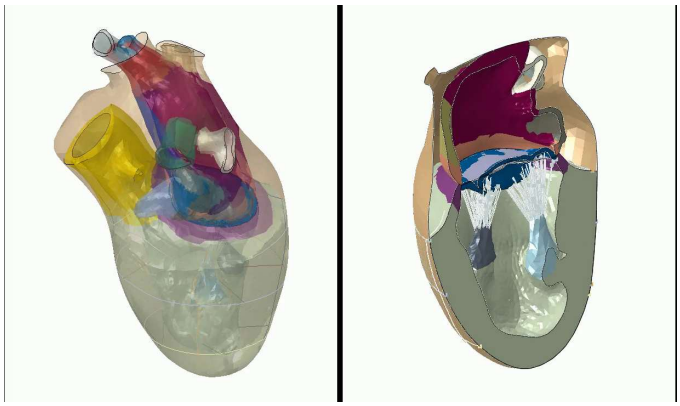
For more information, see *About General Contact in Abaqus/Standard* and **CONTACT CONTROLS ASSIGNMENT*.

No Separation Contact in Abaqus/Explicit

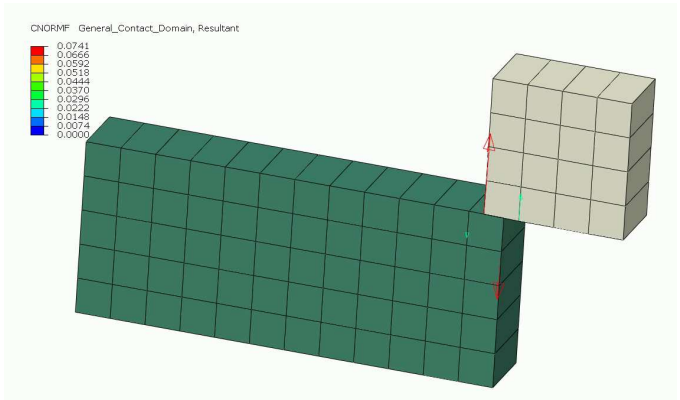
When using general contact in Abaqus/Explicit, you can now define a no separation contact relationship (a contact pressure-overclosure behavior that prevents surfaces from separating in the normal direction once they have come into contact) without the need to use a rough frictional model.

Benefits: You can solve new classes of problems in which surfaces in contact remain in contact but are able to slide tangentially relative to one another.

The animation below shows two views of an Abaqus/Explicit beating heart simulation. As the heart contracts in this simulation, it remains in contact with, and slides relative to, the pericardium (membrane sac). Despite tensile contact stress developing, it represents the effects of fluid within the pericardial cavity (although this fluid is not modeled directly in this simulation). The ability for contact to resist separation as sliding occurs is crucial to obtaining realistic behavior in this simulation. In earlier releases, you could only specify no-separation behavior in combination with “rough” friction for general contact in Abaqus/Explicit.



However, “no separation” contact behavior in the normal direction does not prevent loss of contact for general contact in Abaqus/Explicit when one surface slides off another surface, as shown in the animation below.



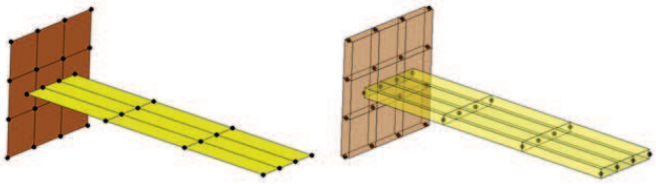
For more information, see *Using the No Separation Relationship* and **SURFACE BEHAVIOR*.

Contact Surface Representation for Shells in Abaqus/Standard

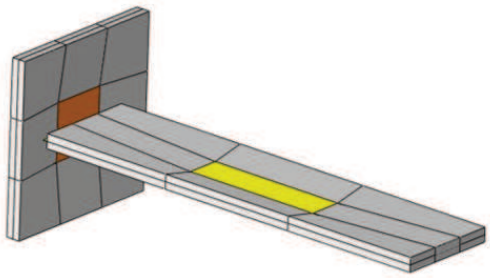
You can specify that Abaqus/Standard internally generate surface elements representing the shell perimeter for contact calculations in general contact.

Benefits: You can more accurately model the shell perimeter.

The default contact surface representation for a part modeled with shell elements is a double-sided shell reference surface that considers the shell thickness and shell offset. The example below shows two bodies meshed with shell elements in contact. The perimeter of one part is in contact with the shell element faces of the other part. The left image shows no thickness rendering, while the right image includes thickness rendering.



Now, you can specify that Abaqus/Standard internally generate surface elements near shell perimeters, as shown in the example below. Surface elements are created right at the shell perimeter, with an additional row of elements extending inward toward the shell interior, ensuring a continuous reference surface representation of the shell. These surface elements can be removed from display in the postprocessor to return to a view such as in the left image above.



For more information, see *About General Contact in Abaqus/Standard* and **CONTACT CONTROLS ASSIGNMENT*.

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 Interactions Guide*.

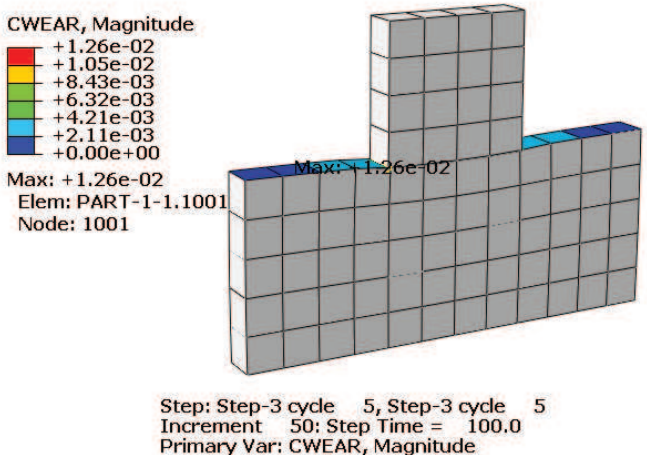
2025 GA

Step Cycling for Contact Wear Simulations

You can now use step cycling and step cycling controls to avoid explicitly specifying each instance of repeated steps associated with wear accumulation and to allow the number of physical wear cycles represented per simulated cycle to evolve based on current wear rates in Abaqus/Standard.

Benefits: Step cycling makes it easier to specify the model and enables more computationally efficient wear simulations.

In many contact wear workflows, surface wear accumulates over thousands of physical wear cycles. It is often prohibitively expensive to simulate each physical wear cycle, so each simulated cycle typically represents a "batch" of multiple similar physical cycles. The animation below shows a wear simulation over five simulated cycles that represent 110,000 physical wear cycles in total. Step cycling allows the "batch size" to evolve during the simulation based on the maximum incremental wear distance in the simulated cycle. The number of simulated cycles is often not known before running such a simulation. For each simulated cycle, the user-specified wear coefficients are scaled by the current batch size.



The animation shows the wear distance accumulation on the slider and the bottom block as the slider repeatedly moves back and forth. In this example, the simulated wear cycle is modeled in two steps:

- In “Step-2” the slider moves to the right and returns to the center.
- In “Step-3” the slider moves to the left and returns to the center.

These two steps are repeated using step cycling until a threshold wear distance exceeds 0.012 or reaches a cap of 10 repetitions of the simulated wear cycle. The batch size for scaling the wear coefficients has an initial value of 10,000 and a cap of 25,000. The target incremental wear distance is 0.006 accumulated per simulated cycle.

The batch size for the next simulated wear cycle is internally calculated so that the estimated maximum wear accumulation is less than the target value of 0.006 while respecting the batch size cap. The example simulation reaches the threshold wear distance in 5 simulated cycles. The batch size for the initial cycle is 10,000 based on user input. For the subsequent cycles, the batch sizes are calculated to be 25,000 (the cap value) while maintaining the maximum wear accumulated in each simulated cycle to be less than 0.006.

For more information, see *Contact Wear*, **STEP CYCLING*, and **STEP CYCLING CONTROL*.

Contact Involving Beams with Noncircular Cross-Sections in Abaqus/Explicit

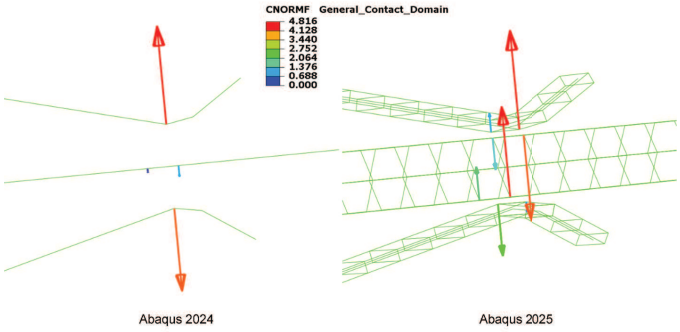
The contact surface representation of beams with noncircular cross-sections in Abaqus/Explicit now uses internally generated meshes of surface elements that are available for postprocessing.

Benefits: The new contact surface representation is consistent with Abaqus/Standard, and it more accurately and realistically represents contact output.

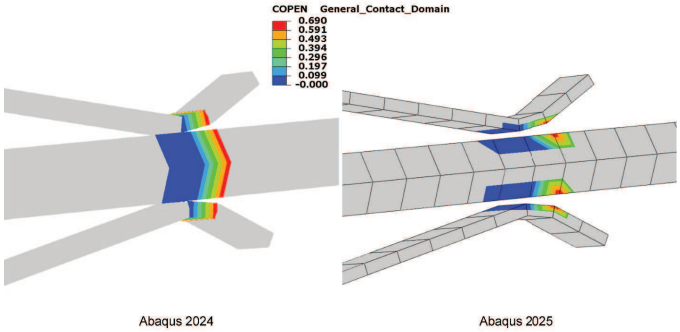
In earlier releases of Abaqus/Explicit, similar internally generated meshes could be used for simulations but they were not available for postprocessing. The images below compare plots of contact forces and contact opening distances on beam elements to plots on internally generated surface-element representations of the beams. Plots on surface elements show variations over the surfaces, similar to plots that would occur on the outer surface of a solid body, whereas plots based on beam elements are limited in that they represent a single value at each axial location.

Furthermore, some rounding of edges in the contact representation to improve numerics is achieved by offsetting nodes of internally generated surface elements inward and adding a small corresponding contact thickness. This contact thickness is not represented in plots, so the parts appear a bit smaller than the actual size.

The following image shows the tweezers beam model contact force vector plot across versions (beam rendering is not available for vector plots):



The following plot shows the tweezers beam model contact opening distance output in previous versions (beam rendering shown for Abaqus 2024):



For more information, see *About General Contact in Abaqus/Explicit*, *About General Contact in Abaqus/Standard*, and **CONTACT CONTROLS ASSIGNMENT*.

2024 FD03

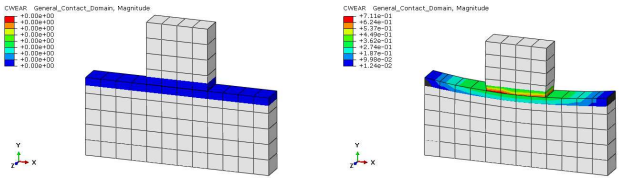
Arbitrary Lagrangian-Eulerian (ALE) Adaptive Meshing Supports Contact Wear in Abaqus/Explicit

You can now use arbitrary Lagrangian-Eulerian (ALE) adaptive meshing together with wear in Abaqus/Explicit, which accounts for local surface wear distances in underlying element calculations.

Benefits: Users can more easily and accurately account for wear distances that are large compared to element dimensions.

The capability to simulate wear in general contact in Abaqus/Explicit was introduced in 2024 FD02. In that release Abaqus/Explicit treated the evolution of wear exclusively by considering nodal wear distances in contact penetration calculations. Now you can use arbitrary Lagrangian-Eulerian (ALE) adaptive meshing together with wear in Abaqus/Explicit to account for surface wear (erosion) in part geometries while preserving mesh quality, even for wear distances on the order of element dimensions.

The simulation on the left below shows the preexisting behavior (wear distances accounted for with offsets in contact penetration calculations). The simulation on the right demonstrates the use of ALE in the context of wear modeling. Similar wear accumulation occurs for both models, but it is much easier to interpret the wear configurations with ALE. Most wear occurs on the smaller block in this model, with the maximum wear distance being approximately half of the original element depth. In this example, ALE mesh smoothing effectively provides a realistic representation of the evolving part geometry while maintaining good mesh quality.



For more information, see *Contact Wear* and **ADAPTIVE MESH*.

2024 FD02

Cohesive Contact in Abaqus/Explicit

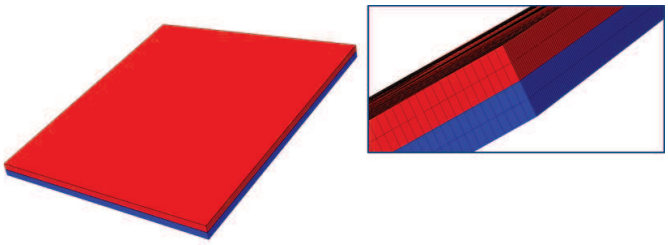
Performance for Abaqus/Explicit models with cohesive contact is improved compared to earlier releases. In addition, scaling the default contact penalty stiffness now also scales the default contact cohesive stiffness in Abaqus/Explicit; this behavior is consistent with that in Abaqus/Standard.

Benefits: The improvements facilitate more efficient user workflows and user adjustments to increase or decrease the contact cohesive stiffness from its default value.

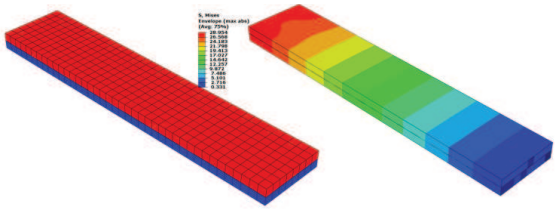
Enhancements to cohesive contact in Abaqus/Explicit make cohesive contact a preferred alternative to surface-based tie constraints for modeling permanent, stiff bonds in more cases (for example, models where ties result in large implicit constraint systems or when you require interfacial traction results across connected surfaces).

Performance improvements for cohesive contact are most noticeable for models with a large fraction of nodes participating in cohesive contact. For example, simulations for the models shown in the images below with permanent, stiff cohesive contact bonds have sped up by 13% and 12%, respectively, compared to simulations in 2024 FD01.

The following figure shows two plates modeled with solid elements bonded together with permanent, stiff cohesive contact:

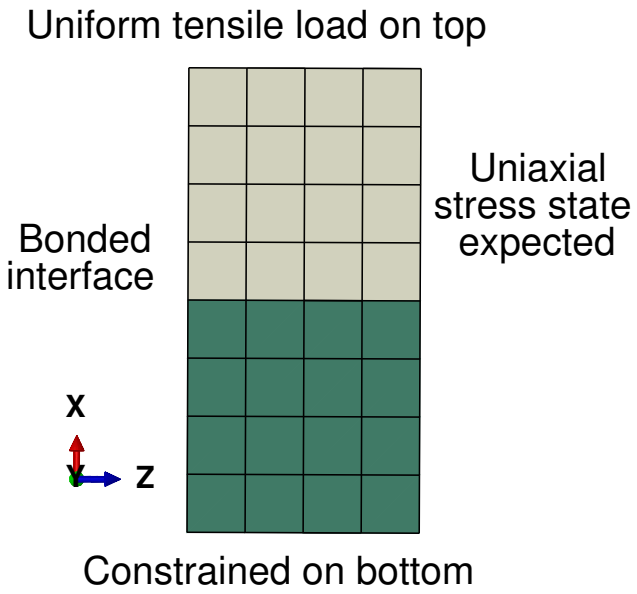


The following figure shows two plates modeled with shell elements (shown with shell thickness rendering) bonded together with permanent, stiff cohesive contact:

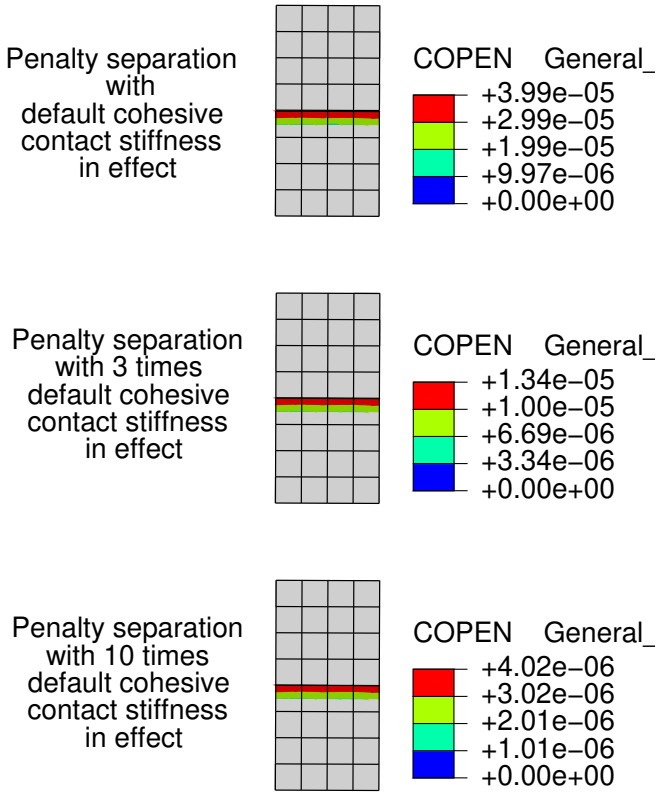


A second enhancement allows the default cohesive stiffness to be scaled by existing options to scale the contact penalty stiffness in Abaqus/Explicit. Cohesive contact, like other contact constraints, has a finite bond stiffness. You can assign the bond stiffness or allow it to take on the same contact penalty stiffness that would be in effect to resist penetrations for regular contact behavior, as discussed in *Contact Cohesive Behavior*. You can use a scale factor to increase the cohesive stiffness relative to the default value.

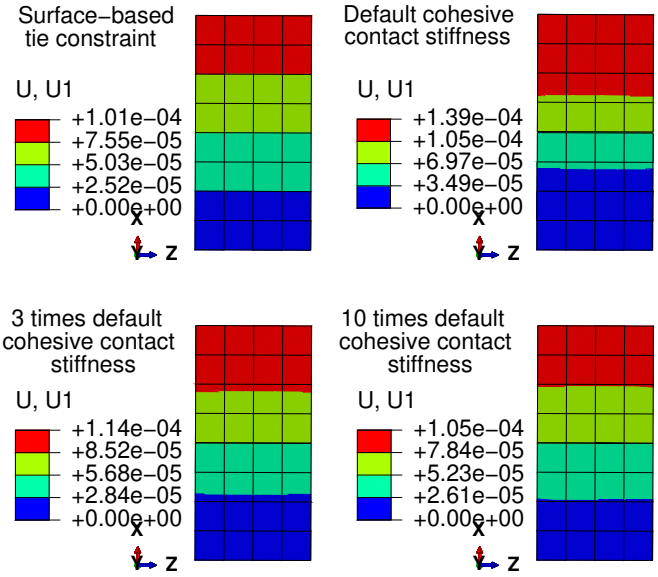
The image below shows two blocks bonded together with uniform tensile (negative pressure) loading of 1×10^6 N/m^2 applied to the top face of the top block while the bottom of the bottom block is constrained. Loading is smoothly ramped on and damping is included to obtain a quasi-static solution.



The image below shows the vertical stress (S11) for three model variants. The axial stress results correspond to the expected uniaxial stress solution regardless of details of how the bond is modeled.



The figure below shows the vertical displacement (U1) for four model variants. Some additional displacement occurs when a penalty stiffness is used to enforce the bond. Increasing the penalty stiffness via a scale factor is a convenient mechanism to control the resistance to relative motion across the bond.



The mesh is quite coarse in this example. The default penalty stiffness would be greater with a more refined mesh. Increasing the penalty stiffness tends to decrease the time increment size in Abaqus/Explicit unless contact mass scaling is used (see *Mass Scaling to Account for Contact Stiffness*).

Alternatively, you can directly specify cohesive stiffness components, as discussed in *Cohesive Stiffness*. When you specify the cohesive stiffness, the penalty stiffness scale factor has no effect.

For more information, see *Contact Cohesive Behavior*, **COHESIVE BEHAVIOR*, **CONTACT CONTROLS ASSIGNMENT*, **CONTACT MASS SCALING*, *Contact Controls for General Contact in Abaqus/Explicit*, and *Mass Scaling*.

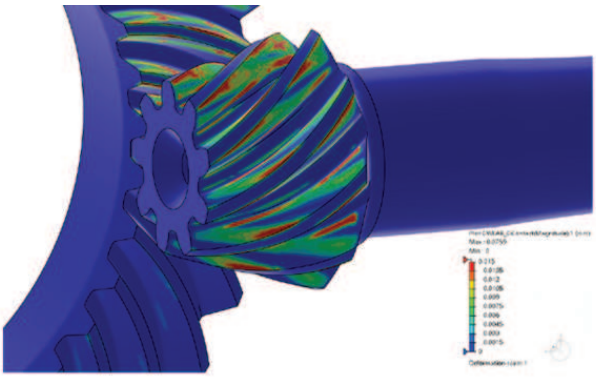
Contact Wear in Abaqus/Explicit

Abaqus/Explicit can now model the evolution of wear and visualize nodal wear distances on contact surfaces due to mechanical contact based on the Archard's wear equation.

Benefits: You can now model wear accumulated at the contact level in Abaqus/Explicit.

This new capability is available in Abaqus/Explicit for general contact analysis involving solid elements. Similar to the implementation introduced for Abaqus/Standard in 2024 FD01, the modeling approach accounts for local surface wear distances in contact penetration calculations but not in underlying element calculations.

The image below shows an example of accumulation of wear (shown as CWEAR contour output) in a simplified helical bevel gear analysis.



For more information, see *Contact Wear*, **SURFACE PROPERTY*, **SURFACE PROPERTY ASSIGNMENT*, and **WEAR SURFACE PROPERTIES*.

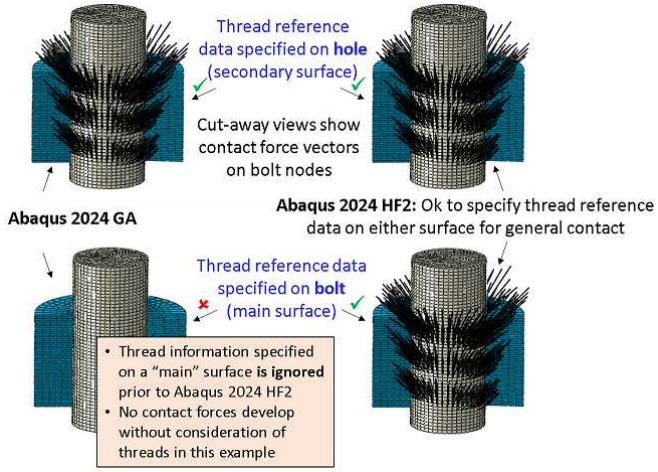
Specifying Reference Thread Geometry in Abaqus/Standard

Abaqus/Standard allows reference thread geometry to be specified on either surface (secondary or main) of a small-sliding interaction for general contact.

Benefits: Reference thread data for reference thread geometry can be specified for either surface of a small-sliding interaction of general contact.

Allowing reference thread geometry to be specified on either surface of a small-sliding interaction for general contact improves usability. Previously, this data was expected to be specified on whichever surface acting as the secondary surface of the contact formulation. However, main and secondary roles are typically left as default assignments for general contact in Abaqus/Standard, so a user typically would not know the secondary and main roles before running a simulation (and typically should not need to be aware of those roles).

The images below show cut-away views of the contact force vectors on the bolt nodes.The right side shows that results in Abaqus 2024 HF2 are insensitive to whether reference thread geometry is specified for the bolt or hole surface for small sliding interactions of general contact. In prior versions, the expected behavior was obtained only if reference thread geometry was specified for the secondary surface.



2024 FD01

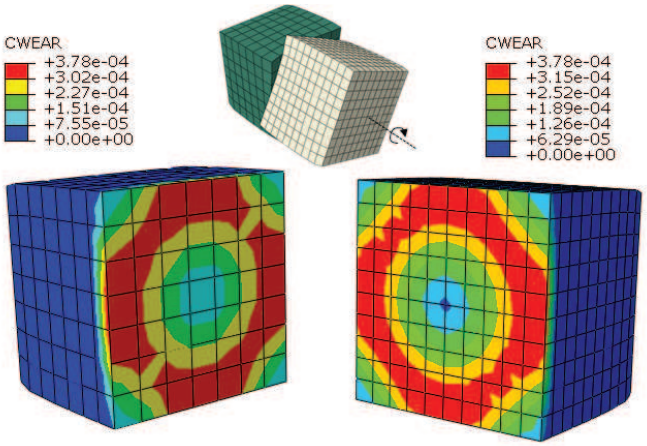
Contact Wear in Abaqus/Standard

You can now model the evolution of wear and visualize nodal wear distances on contact surfaces due to mechanical contact, based on the Archard's wear equation.

Benefits: Abaqus/Standard can now handle wear modeling workflows in automotive, aerospace, life sciences, and packaging applications.

In many workflows, mechanical contact can cause wear on contacting surfaces due to mechanisms like abrasion and fretting. The new wear capability in Abaqus/Standard models wear evolution based on the Archard's wear equation. Accordingly, contact nodal wear distances evolve based on wear properties of the surface material and on contact variables such as normal stress and slip distance. Abaqus/Standard treats wear distances as nodal offsets to the contact algorithm, resulting in a simplified approach to model wear fully within contact. It does not model the changes to geometry from wear explicitly. The approach considers the two-way physical influence of wear distance evolution on contact stresses (and underlying material stresses) and vice versa.

The figures below shows the wear accumulation on the contact surfaces for two blocks in contact.



Wear properties for the Archard's wear equation are specified on the contact surfaces. Nodal wear distances accumulate as the blocks slide relative to each other under compression. The nodal wear distances on the two surfaces are shown as CWEAR output contours at the end of the simulation.

For more information, see *Contact Wear*, *Contact wear modeling*, **SURFACE PROPERTY*, **SURFACE PROPERTY ASSIGNMENT*, and **WEAR SURFACE PROPERTIES*.

Defining Contact Interactions

In this section:

- *About Contact Interactions*
- *Defining General Contact in Abaqus/Standard*
- *Defining Contact Pairs in Abaqus/Standard*
- *Defining General Contact in Abaqus/Explicit*
- *Defining Contact Pairs in Abaqus/Explicit*

About Contact Interactions

Overview

This section presents an overview of the contact analysis capabilities in Abaqus.

References:

- *Contact Elements in Abaqus/Standard*
- *Defining General Contact in Abaqus/Explicit*
- *Defining Contact Pairs in Abaqus/Explicit*

Available Contact Algorithms in Abaqus

Abaqus provides more than one approach for defining contact. Abaqus/Standard includes the following approaches for defining contact:

- general contact;
- contact pairs; and
- contact elements.

Abaqus/Explicit includes the following approaches for defining contact:

- general contact; and
- contact pairs.

Each approach has somewhat unique advantages and limitations.

The remainder of this section is organized as follows:

- first, discuss common aspects of the surface-based contact-definition approaches (that is, contact pairs and general contact);
- next, provide an overview of the contact definition approaches in Abaqus/Standard and the contact definition approaches in Abaqus/Explicit;
- finally, discuss compatibility between the contact algorithms in Abaqus/Standard and Abaqus/Explicit.

Defining a Surface-Based Contact Simulation

A contact simulation using contact pairs or general contact is defined by specifying:

- surface definitions for the bodies that could potentially be in contact;
- the surfaces that interact with one another (the contact interactions);
- any nondefault surface properties to be considered in the contact interactions;
- the mechanical and thermal contact property models, such as the pressure-overclosure relationship, the friction coefficient, or the contact conduction coefficient;
- any nondefault aspects of the contact formulation; and

- any algorithmic contact controls for the analysis.

In many cases you do not need to explicitly specify many of the aspects listed above because the default settings are usually appropriate.

Surfaces

Surfaces can be defined at the beginning of a simulation or upon restart as part of the model definition (see *About Surfaces*). Abaqus has five classifications of contact surfaces:

- element-based deformable and rigid surfaces (*Element-Based Surface Definition*);
- node-based deformable and rigid surfaces (*Node-Based Surface Definition*);
- analytical rigid surfaces (*Analytical Rigid Surface Definition*);
- Eulerian material surfaces for Abaqus/Explicit (*Eulerian Surface Definition*); and
- crack surfaces for enriched elements (*Defining a Crack Surface*).

Surfaces of the same type can be combined to create new surfaces (see *Operating on Surfaces*). However, with regard to contact a combined surface can be used only with general contact in Abaqus/Explicit.

When the general contact algorithm is used, Abaqus also provides a default all-inclusive, automatically defined surface that includes all element-based surface facets (in Abaqus/Standard and in Abaqus/Explicit), all crack surfaces for enriched elements (in Abaqus/Standard only), all analytical rigid surfaces (in Abaqus/Explicit only), and all Eulerian materials (in Abaqus/Explicit only) in the model.

Contact Interactions

Contact interactions for contact pairs and general contact are defined by specifying surface pairings and self-contact surfaces. General contact interactions typically are defined by specifying self-contact for the default surface, which allows an easy, yet powerful, definition of contact. (Self-contact for a surface that spans multiple bodies implies self-contact for each body as well as contact between the bodies.)

At least one surface in an interaction must be a non-node-based surface, and at least one surface in an interaction must be a non-analytical rigid surface. Additional restrictions and guidelines for contact surfaces are discussed for each contact definition approach. The definition of contact pairs is discussed in detail in *About Contact Pairs in Abaqus/Standard* and *About Contact Pairs in Abaqus/Explicit*. The definition of general contact interactions is discussed in detail in *About General Contact in Abaqus/Standard* and *About General Contact in Abaqus/Explicit*.

Surface Properties

Nondefault surface properties (such as thickness and, in some cases, offset) can be defined for particular surfaces in a contact model. In addition, you can control which edges of a surface will be included in the general contact domain in Abaqus/Explicit. Surface properties for contact pairs are discussed in *Assigning Surface Properties for Contact Pairs in Abaqus/Standard* and *Assigning Surface Properties for Contact Pairs in Abaqus/Explicit*. Surface properties for general contact are discussed in *Surface Properties for General Contact in Abaqus/Standard* and *Assigning Surface Properties for General Contact in Abaqus/Explicit*.

Contact Properties

Contact interactions in a model can refer to a contact property definition, in much the same way that elements refer to an element property definition. By default, the surfaces interact (have constraints) only in the normal direction to resist penetration. The other mechanical contact interaction models available depend on the contact algorithm and whether Abaqus/Standard or Abaqus/Explicit is used (see *About Mechanical Contact Properties*). Some of the available models are:

- softened contact (*Contact Pressure-Overclosure Relationships* and *Frictional Behavior*),
- contact damping (*Contact Damping* and *Frictional Behavior*),
- friction (*Frictional Behavior*), and
- a user-defined constitutive model for surface interactions (*User-Defined Interfacial Constitutive Behavior*).

The thermal, thermal-electrical, and pore-fluid surface interaction models available in Abaqus are discussed in *Thermal Contact Properties*, *Electrical Contact Properties*, and *Pore Fluid Contact Properties*, respectively.

Contact interaction models are defined as model data except for contact pairs in Abaqus/Explicit, in which case they are defined as history data. Information on assigning contact properties to contact pairs can be found in *Assigning Contact Properties for Contact Pairs in Abaqus/Standard* and *Assigning Contact Properties for Contact Pairs in Abaqus/Explicit*. Information on assigning contact properties to general contact interactions can be found in *Contact Properties for General Contact in Abaqus/Standard* and *Assigning Contact Properties for General Contact in Abaqus/Explicit*.

The crush stress associated with the CZone analysis capability is specified as a material property and has the effect of limiting the contact stress under specific circumstances discussed in *CZone Analysis*.

Numerical Controls

The default algorithmic controls for contact analyses are usually sufficient, but you can adjust numerical controls for some special cases. For example, depending on the contact algorithm used, the numerical controls for the contact formulation, the main and secondary roles for the contact surfaces, and the sliding formulation are provided. Information on contact formulations and numerical methods used by the contact algorithms is provided in *Contact Formulations in Abaqus/Standard* and *Contact Formulations for Contact Pairs in Abaqus/Explicit*. The available numerical controls for the various contact algorithms are discussed in *Contact Controls Specific to General Contact in Abaqus/Standard*, *Generally Applicable Contact Controls in Abaqus/Standard*, *Contact Controls for General Contact in Abaqus/Explicit*, and *Contact Controls for Contact Pairs in Abaqus/Explicit*.

Contact-Dependent Loading

The following loading-related capabilities are coupled to Abaqus contact capabilities such that loading depends on local contact state:

- Fluid pressure penetration (*Fluid Pressure Penetration Loads*)
- Surface-based convection loading (*Specifying Surface-Based Film Conditions*)
- Surface-based radiation loading to ambient (*Specifying Surface-Based Radiation to Ambient*)

Contact Simulation Capabilities in Abaqus/Standard

Abaqus/Standard provides the following approaches for defining contact interactions: general contact, contact pairs, and contact elements. Contact pairs and general contact both use surfaces to define contact; comparisons of these approaches are provided later in this section. Contact elements are provided for certain interactions that cannot be modeled with either general contact or contact pairs; however, it is generally recommended to use general contact or contact pairs if possible.

Capabilities of Contact Pairs and General Contact in Abaqus/Standard

Contact pairs and general contact combine to provide the following capabilities in Abaqus/Standard:

- Contact between two deformable bodies. The structures can be either two- or three-dimensional, and they can undergo either small or finite sliding. Examples of such problems include the assembly of a cylinder head gasket and the slipping between two components of a threaded connector.

- Contact between a rigid surface and a deformable body. The structures can be either two- or three-dimensional, and they can undergo either small or finite sliding. Examples of such problems include metal forming simulations and analyses of rubber seals being compressed between two components.
- Finite-sliding self-contact of a single deformable body. An example of such a problem is a complex rubber seal that folds over on itself.
- Small-sliding or finite-sliding interaction between a set of points and a rigid surface. These models can be either two- or three-dimensional. An example of this type of problem is the pull-in of an underwater cable that is resting on the seabed, with the seabed modeled as a rigid surface.
- Contact between a set of points and a deformable surface. These models can be either two- or three-dimensional. An example of this class of contact problem is the design of a bearing where one of the bearing surfaces is modeled with substructures.
- Problems where two separate surfaces need to be "tied" together so that there is no relative motion between them. This modeling technique allows for joining dissimilar meshes.
- Coupled thermomechanical interaction between deformable bodies with finite relative motion. The analysis of a disc brake is an example of such a problem.
- Coupled thermal-electrical-structural interaction between deformable bodies with finite relative motion. An example of this type of problem is the analysis of resistance spot welding.
- Coupled pore fluid-mechanical interaction between bodies. An example of this type of problem is the analysis of the interfaces between layered soil material at a waste disposal site.

Coupled thermomechanical and coupled thermal-electrical-structural interactions can be included in any of the above examples as long as both of the surfaces are deformable.

Choosing between General Contact or Contact Pairs in Abaqus/Standard

For most contact problems you have a choice of whether to define contact interactions using general contact or contact pairs. In Abaqus/Standard the distinction between general contact and contact pairs lies primarily in the user interface, the default numerical settings, and the available options. The general contact and contact pair implementations share many underlying algorithms.

The contact interaction domain, contact properties, and surface attributes are specified independently for general contact, offering a more flexible way to add detail incrementally to a model. The simple interface for specifying general contact allows for a highly automated contact definition; however, it is also possible to define contact with the general contact interface to mimic traditional contact pairs. Conversely, specifying self-contact of a surface spanning multiple bodies with the contact pair user interface (if the surface-to-surface formulation is used) mimics the highly automated approach often used for general contact.

In Abaqus/Standard traditional pairwise specifications of contact interactions may result in more efficient analyses as compared to an all-inclusive self-contact approach to defining contact. Therefore, there is often a trade-off between ease of defining contact and analysis performance. Abaqus/CAE provides a contact detection tool that greatly simplifies the process of creating traditional contact pairs for Abaqus/Standard (see *Understanding contact and constraint detection*).

Default Settings for General Contact and Contact Pairs

Differences in default settings for general contact and contact pairs in Abaqus/Standard include the following:

- Contact formulation: By default, general contact uses the finite-sliding, surface-to-surface formulation supplemented by the finite-sliding, edge-to-surface; finite-sliding, edge-to-edge; and finite-sliding, vertex-to-surface formulations. For crack surfaces on enriched elements, edges and vertices are deactivated. In addition, you can specify the small-sliding tracking approach over portions of the general contact domain. Edges and vertices are deactivated over regions where small sliding is active. Contact pairs use the

finite-sliding, node-to-surface formulation by default except when the contact detection tool in Abaqus/CAE is used to create the contact pairs, in which case the finite-sliding, surface-to-surface formulation is used by default. See *Contact Formulations in Abaqus/Standard* for a discussion of contact formulations.

- Treatment of shell thickness and offset: General contact automatically accounts for thicknesses and offsets associated with shell-like surfaces regardless of the finite-sliding (default) or the small-sliding tracking approaches. Contact pairs that use the finite-sliding, node-to-surface formulation do not account for shell thicknesses and offsets. See *Surface Properties for General Contact in Abaqus/Standard* and *Assigning Surface Properties for Contact Pairs in Abaqus/Standard* for discussions of surface properties for contact in Abaqus/Standard.
- Contact constraint enforcement: General contact uses the penalty method to enforce the contact constraints by default. Contact pairs that use the finite-sliding, node-to-surface formulation use a Lagrange multiplier method to enforce contact constraints by default in most cases. See *Contact Constraint Enforcement Methods in Abaqus/Standard* for a discussion of contact constraint enforcement methods.
- Treatment of initial overclosures: General contact eliminates initial overclosures with strain-free adjustments by default. Contact pairs treat initial overclosures as interference fits to be resolved in the first increment of the analysis by default. See *Contact Initialization for General Contact in Abaqus/Standard*, *Modeling Contact Interference Fits in Abaqus/Standard*, and *Contact Initialization for Contact Pairs in Abaqus/Standard* for more information on contact initialization in Abaqus/Standard.
- Main-secondary assignments: General contact automatically assigns pure main and secondary roles for most contact interactions and automatically assigns balanced main-secondary roles to other contact interactions. Contact between opposing surfaces of a crack on enriched elements supports only the balanced main-secondary approach. The user must assign main and secondary roles for most contact pairs. See *Contact Controls Specific to General Contact in Abaqus/Standard* and *Choosing the Main and Secondary Surface Roles in a Two-Surface Contact Pair* for discussions of main and secondary roles for contacting surfaces.

Capabilities Available Only for General Contact in Abaqus/Standard

The following capabilities are available only for general contact in Abaqus/Standard (they are not available for contact pairs in Abaqus/Standard):

- Edge-to-surface, edge-to-edge, and vertex-to-surface contact formulations.
- Finite-sliding contact involving XFEM (*Modeling Discontinuities as an Enriched Feature Using the Extended Finite Element Method*) surfaces.
- Automatic assignment of main and secondary roles.

Capabilities Available Only for Contact Pairs in Abaqus/Standard

The following capabilities are available only for contact pairs in Abaqus/Standard (they are not available for general contact in Abaqus/Standard):

- Contact involving rigid surfaces defined with user subroutine *RSURFU*.
- The finite-sliding, node-to-surface contact formulation or a node-based surface specified with the finite-sliding, surface-to-surface contact formulation (the latter automatically reverts to the finite-sliding, node-to-surface contact formulation).
- Interactions using small-sliding approximations where the secondary surface is node based or if the secondary is made of structural elements like beams, pipes, trusses and frames, tied contact.
- Debonding.
- Local definitions of some numerical contact controls.
- Symmetric model generation.

A single analysis can include general contact and contact pair definitions. General contact automatically avoids processing contact interactions that are treated by contact pairs.

Using Contact Elements in Contact Simulations

Abaqus/Standard provides a library of contact elements that can be used to model certain classes of problems. Examples of such problems are:

- Contact interaction between two pipelines or tubes modeled with pipe, beam, or truss elements where one pipe lies inside the other (such as a J-tube pull in offshore piping installation) or the pipes lie next to each other (available in both two and three dimensions; see *Tube-to-Tube Contact Elements*). Alternatively, this category of problems can be simulated using general contact with the beam-to-beam formulation.
- Contact between two nodes along a fixed direction in space. An example of such a problem is the interaction of a piping system with its supports (see *Gap Contact Elements*).
- Simulations using axisymmetric elements with asymmetric deformations, CAXA*n* and SAXA*n* elements. See *Contact Modeling If Asymmetric-Axisymmetric Elements Are Present* for details.
- Heat transfer analyses where the heat flow is one-dimensional. An example of such a problem is the heat flow in a piping system that is discontinuous. The thermal interaction in this problem is one-dimensional, so no surfaces can be defined (see *Gap Contact Elements*).

Defining a Contact Simulation Using Contact Elements

The steps required for defining a contact simulation using contact elements are similar to those needed when defining a surface-based contact simulation:

- create the contact elements or slide lines;
- assign element section properties to the contact elements;
- associate sets of contact elements with the slide lines if applicable; and
- define the contact property models for the contact elements.

The first three steps are discussed in *Contact Elements in Abaqus/Standard* in the sections for each type of contact element. The contact property models for contact elements are identical to those used for surface-based contact.

Contact Simulation Capabilities in Abaqus/Explicit

Abaqus/Explicit provides two algorithms for modeling contact interactions. General contact generally provides much better performance and also has advantages of robustness, with easier model setup for large assemblies and fewer restrictions on types of surfaces involved in contact. The general contact and contact pair algorithms in Abaqus/Explicit differ by more than the user interface; in general, they use completely separate implementations with many key differences in the designs of the numerical algorithms.

The two contact algorithms combine to provide the following capabilities in Abaqus/Explicit:

- Contact between rigid and/or deformable bodies.
- Contact of a body with itself.
- Finite-sliding or small-sliding contact.
- Contact with eroding bodies (due to element failure). A node-based surface must be used to model the eroding body if contact pairs are used. General contact allows element-based surfaces to be defined on eroding bodies, so contact between any number of eroding bodies can be modeled.
- General constitutive models for the contact behavior, including user-defined models through user subroutines, relating constraint pressure and shear traction to penetration distance and relative tangential motion.

- Thermal interaction at the surface of a body; for example, conductive heat transfer.
- Contact between Eulerian material and Lagrangian bodies.
- Contact between DEM or SPH particles and other Lagrangian surfaces.
- Contact among DEM particles.
- CZone analysis to model crushing of laminated composites due to contact.
- A friction coefficient defined in terms of average surface temperature and/or field variables.

Choosing between General Contact or Contact Pairs in Abaqus/Explicit

Contact definitions are not entirely automatic with the general contact algorithm but are greatly simplified. The generality of this algorithm is primarily in the relaxed restrictions on the surfaces that can be used in contact. The general contact algorithm in Abaqus/Explicit allows the following (none of which are allowed with the contact pair algorithm in Abaqus/Explicit):

- A surface can span unattached bodies.
- More than two surface facets can share a common edge (allowing "T-intersections" in shells, for example).
- A surface can include deformable and rigid regions; furthermore, the rigid regions need not be from the same rigid body.
- A surface can have mixed parent element types; for example, adjacent surface facets can be on shell and solid elements.
- A surface can be based on combinations of surfaces of the same type.
- An element-based surface can be defined on the interior of solid bodies for use in modeling erosion due to element failure.
- A surface can be defined on the exterior of an Eulerian material instance (see *Eulerian Surface Definition*).

Other benefits of the general contact algorithm in Abaqus/Explicit include the following:

- The general contact algorithm does not place any restrictions on the domain decomposition for domain level parallelization (see *Parallel Execution in Abaqus/Explicit*). The contact pair algorithm significantly degrades parallel performance in most cases.
- The general contact algorithm can enforce edge-to-edge contact for geometric feature edges, perimeter edges of structural elements, and edges defined by beam and truss elements, unlike the contact pair algorithm.
- The general contact algorithm is the only option for enforcing contact between Eulerian materials and Lagrangian bodies (see *Interactions*).
- The general contact algorithm is the only option for enforcing contact involving DEM or SPH particles (see *Discrete Element Method* and *Smoothed Particle Hydrodynamics*).
- General contact can be used to simulate crushing of laminated composites in a CZone analysis (see *CZone Analysis*). With this approach, contact, shell element and material modeling aspects are integrated to model crush fronts at an impact location with less mesh dependence than other damage and failure techniques.
- The general contact algorithm eliminates problematic, nonphysical "bull-nose" extensions that may arise at shell surface perimeters in the contact pair algorithm.
- With the general contact algorithm each secondary node can see contact with multiple facets per increment; with the contact pair algorithm each secondary node can see contact with only one facet per increment unless multiple surface pairings are specified. Likewise, each contact edge can see contact with multiple edges per increment when the general contact algorithm is used.
- The general contact algorithm has some built-in smoothing for element-based surfaces that can be beneficial for modeling contact near corners.

- The general contact algorithm, unlike the contact pair algorithm, removes contact faces and contact edges from the contact domain and, if an interior surface is defined, activates newly exposed surface faces as elements fail. Thus, element-based surfaces can be used to describe eroding solids. This allows contact between multiple eroding solids to be modeled since a node-based surface does not need to be defined on the eroding solid.
- Contact state information (such as the proper contact normal orientation for double-sided surfaces) is transferred across step boundaries in the general contact algorithm even if the contact domain is modified; in the contact pair algorithm, contact state information is transferred across step boundaries only for contact pairs with no modifications.
- The contact interaction domain, contact properties, and surface attributes are specified independently for the general contact algorithm, offering a more flexible way to add detail incrementally to a model.
- The general contact algorithm, unlike the penalty enforcement method in the contact pair algorithm, does not place any restrictions on the domain decomposition parallelization for the thickness and midsurface offset of shells and membranes.
- The general contact algorithm in Abaqus/Explicit has been developed to minimize the need for algorithmic controls.

See *Knee bolster impact with general contact*, *Crimp forming with general contact*, and *Collapse of a stack of blocks with general contact* for example analyses that use the general contact algorithm.

Although the general contact algorithm is more powerful and allows for simpler contact definitions, the contact pair algorithm must be used in certain cases where more specialized contact features are desired. The following features are available in Abaqus/Explicit only when the contact pair algorithm is used:

- Kinematically enforced contact (see *Contact Constraint Enforcement Methods in Abaqus/Explicit*; the general contact algorithm uses only penalty enforcement)
- Small-sliding contact (see *Contact Formulations for Contact Pairs in Abaqus/Explicit*)

The two contact algorithms can be used together in the same Abaqus/Explicit analysis. The general contact algorithm automatically avoids processing interactions that are treated by the contact pair algorithm.

Compatibility between Abaqus/Standard and Abaqus/Explicit

There are fundamental differences in the mechanical contact algorithms in Abaqus/Standard and Abaqus/Explicit even though the input syntax is similar. The main differences are the following:

- Contact pair and general contact definitions in Abaqus/Standard are model definition data (although contact pairs can be removed for a portion of the analysis and added back to the model in a later step of the analysis, as discussed in *Removing and Reactivating Contact Pairs*). In the contact pair algorithm in Abaqus/Explicit contact constraints are history definition data (see *Abaqus Model Definition*); in the general contact algorithm in Abaqus/Explicit contact definitions can be either model or history data.
- Abaqus/Standard typically uses a pure main-secondary relationship for the contact constraints; whereas Abaqus/Explicit typically uses balanced main-secondary contact by default. This difference is primarily due to overconstraint issues unique to Abaqus/Standard.
- The contact formulations in Abaqus/Standard and Abaqus/Explicit differ in many respects due to different convergence, performance, and numerical requirements:
  - Abaqus/Standard provides surface-to-surface, edge-to-surface, and vertex-to-surface formulations, which Abaqus/Explicit does not;
  - Abaqus/Standard and Abaqus/Explicit both provide node-to-surface formulations, but some details associated with surface smoothing, etc. differ in the respective implementations.

- The constraint enforcement methods in Abaqus/Standard and Abaqus/Explicit differ in some respects. For example, both analysis codes provide penalty constraint methods, but the default penalty stiffnesses differ (this is primarily due to the effect of the penalty stiffness on the stable time increment for Abaqus/Explicit).
- The small-sliding contact capability in Abaqus/Standard transfers the load to the main nodes according to the current position of the secondary node, but the small-sliding contact capability in Abaqus/Explicit always transfers the load through the anchor point due to a numerical limitation associated with the implementation.
- Abaqus/Explicit can account for the thickness and midsurface offset of shells and membranes in the contact penetration calculations (although in some cases changes in the thickness upon deformation are not accounted for in the contact calculations). Abaqus/Standard cannot account for the thickness and offset of shells and membranes when using the finite-sliding, node-to-surface contact formulation (but can account for the original thickness and offset in all other contact formulations).

As a result of these differences, contact definitions specified in an Abaqus/Standard analysis cannot be imported into an Abaqus/Explicit analysis and vice versa (see *Transferring Results between Abaqus/Explicit and Abaqus/Standard*). However, in many cases you can successfully respecify a contact definition in an import analysis.

Defining General Contact in Abaqus/Standard

In this section:

- *About General Contact in Abaqus/Standard*
- *Surface Properties for General Contact in Abaqus/Standard*
- *Contact Properties for General Contact in Abaqus/Standard*
- *Stabilization for General Contact in Abaqus/Standard*

About General Contact in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Contact Interactions*
- *Contact Property Models*
- **CONTACT*
- **CONTACT INCLUSIONS*
- **CONTACT EXCLUSIONS*
- *Defining general contact*

Overview

Abaqus/Standard provides two algorithms for modeling contact and interaction problems: the general contact algorithm and the contact pair algorithm.
See *About Contact Interactions* for a comparison of the two algorithms. This section describes how to include general contact in an Abaqus/Standard analysis, how to specify the regions of the model involved in general contact interactions, and how to obtain output from a general contact analysis.

The general contact algorithm in Abaqus/Standard:

- is specified as part of the model definition;
- allows very simple definitions of contact with very few restrictions on the types of surfaces involved;
- uses sophisticated tracking algorithms to ensure that proper contact conditions are enforced efficiently;
- can be used simultaneously with the contact pair algorithm (that is, some interactions can be modeled with the general contact algorithm, while others are modeled with the contact pair algorithm);
- can be used with two- or three-dimensional surfaces; and
- by default, uses the finite-sliding, surface-to-surface contact formulation as the primary contact formulation, supplemented by the edge-to-surface, edge-to-edge, and vertex-to-surface contact formulations. You can also specify a small-sliding tracking approach over portions or the entire general contact domain.

Defining a General Contact Interaction

The definition of a general contact interaction consists of specifying:

- the general contact algorithm and defining the contact domain (that is, the surfaces that interact with one another), as described in this section;
- the contact surface properties (*Surface Properties for General Contact in Abaqus/Standard*);
- the mechanical contact property models (*Contact Properties for General Contact in Abaqus/Standard*);
- the controls associated with the initial contact state (*Contact Initialization for General Contact in Abaqus/Standard*); and
- the algorithmic contact controls (*Contact Controls Specific to General Contact in Abaqus/Standard*).

See *Impact analysis of a pawl-ratchet device* for an example of an analysis that uses general contact to define contact between the various components of an assembly.

Surfaces Used for General Contact

The general contact algorithm in Abaqus/Standard allows for quite general characteristics in the surfaces that it uses, as discussed in *About Contact Interactions*. For detailed information on defining surfaces in Abaqus/Standard for use with the general contact algorithm, see *Element-Based Surface Definition*.

In addition, Abaqus/Standard automatically defines an all-inclusive surface that is convenient for prescribing the contact domain, as discussed later in this section. The all-inclusive automatically defined surface includes all element-based surface facets, crack surfaces for enriched elements, and analytical rigid surfaces.

The general contact algorithm does not consider contact involving node-based surfaces, although Abaqus/Standard can include these surfaces in contact pairs in analyses that also use general contact.

Contact Surface Representation for Beams

Figure 1 shows contact surface representations for beams with various cross-sections based on surface element meshes that are internally generated by Abaqus for general contact. *Figure 2* shows beam element rendering for the corresponding beam meshes. By default, Abaqus creates the internally generated surface element meshes for general contact except for circular beam cross-sections. You can optionally specify the surface-element representation for circular cross-section beams. Abaqus bases the default contact representation for a circular cross-section beam on beam nodes and contact edges associated with the beam element reference lines, with the contact thickness corresponding to the beam radius.

Nodal motion for nodes of internally generated surface elements is dictated by the translations and rotations of the beam nodes. Most of the internally generated surface elements shown in *Figure 1* are double-sided and have nonzero thickness. Contact representations of exterior corners for internally generated surface-element meshes include rounding of exterior corners, although this rounding is not rendered during postprocessing.

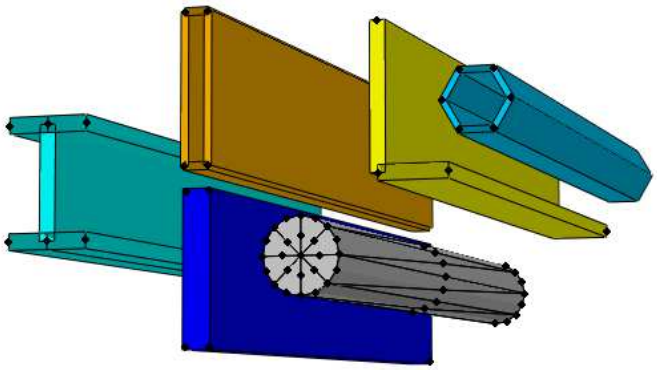


Figure 1: Internally generated surface representations of various beam cross-sections.

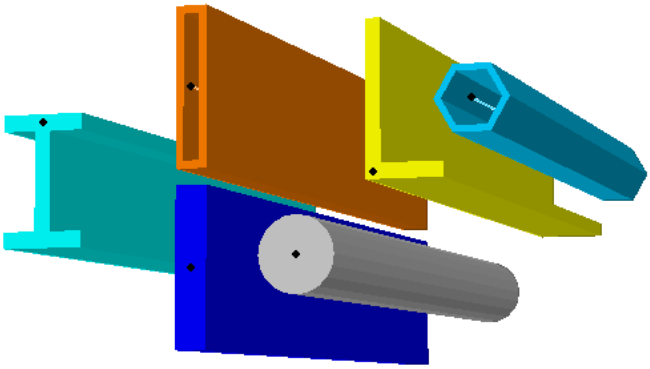


Figure 2: Beam element rendering of various beam cross-sections.

The use of internally generated surface-element meshes for contact involving beams leads to contact faces being part of the contact surface representation, which enables:

- Additional contact formulations such as the surface-to-surface contact formulation to participate in contact enforcement.
- Graphical representation of circumferential variations of contact output fields on beam surfaces. For example, *Figure 3* shows contact pressure results for a hexagonal cross-section beam pinched between two rectangular cross-section beams. For the hexagonal beam, each original beam node has a corresponding ring of six internally generated surface-element nodes with independent CSTRESS values.

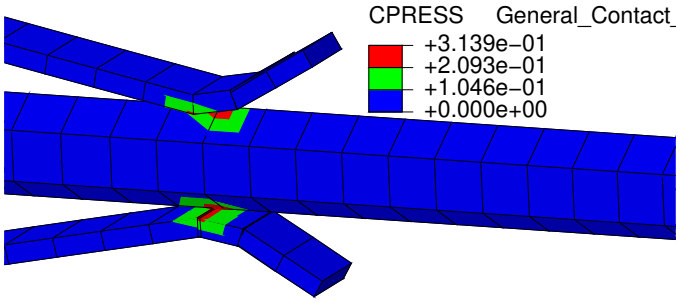


Figure 3: Example of contact pressure (CSTRESS) results for beams with contact surfaces represented by internally generated surface elements.

Some rounding of beam edges in the contact representation (which is beneficial for contact numerics) is achieved by offsetting the nodal positions of the internally generated surface-element meshes inward from the vertex locations of the cross-section at the axial locations of the beam nodes. General contact introduces a corresponding surface thickness automatically, such that the contact representation is as shown in *Figure 4* for a rectangular cross-section. Although the surface thickness is not represented on surface elements during postprocessing, this thickness is accounted for in the contact calculations. Therefore, the contact surface representation appears slightly smaller during postprocessing than the actual beam cross-section.

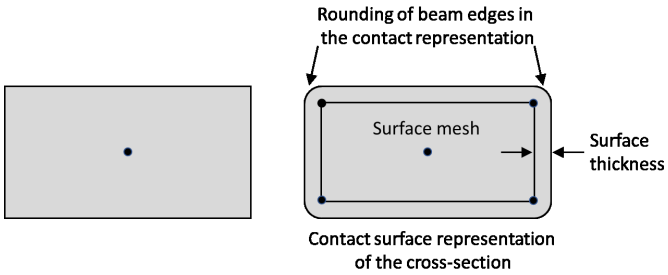


Figure 4: Contact surface representation of a rectangular beam cross-section.

Directly assigning a nonzero surface thickness for contact (see *Surface Thickness*) to a beam with a noncircular cross-section will contribute additional contact surface thickness to the contact representation of the beam, as shown in *Figure 5*.

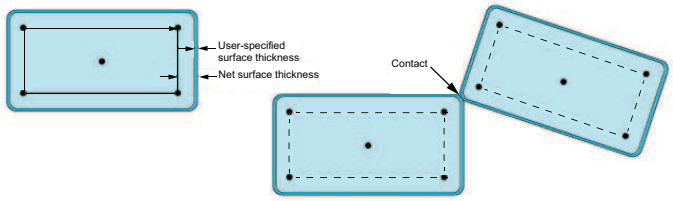


Figure 5: Effect of user-specified surface thickness for a rectangular beam cross-section.

Motion of each surface-element node is driven by the corresponding beam node via an internally generated beam MPC (see *Using MPC Type BEAM*). Abaqus redistributes the contact forces acting on these nodes to the original beam nodes as forces and moments to provide the correct beam response. The surface element-based surfaces inherit the associated general contact property assignments from the corresponding beam surfaces. Meshes of surface elements for beam cross-sections are available for postprocessing (see *Output for Beams with Actual Beam Cross-Section Representation and Shells with Actual Shell Perimeter Representation*).

A beam represented by reference-line edges and nodes of beam elements for contact can participate in general contact in Abaqus/Standard in the following ways:

- Qualifying nodes of beam elements can participate in vertex-to-surface contact.
- Edges of beam elements can participate in edge-to-edge, edge-to-surface, and possibly surface-to-beam contact.

A beam represented by internally generated surface elements for contact can participate in general contact in Abaqus/Standard in the following ways:

- Qualifying nodes of surface elements can participate in vertex-to-surface contact.
- Feature edges of surface elements can participate in edge-to-edge contact and edge-to-surface contact.
- Faces of surface elements can participate in vertex-to-surface, edge-to-surface, surface-to-surface, and possibly surface-to-beam contact.

Alternatively, the contact surface geometry of a noncircular beam cross-section can be approximated with a circular cross-section encompassing the actual cross-section. *Figure 6* shows a rendering of beam elements with a rectangular cross-section (left) and the optional, nondefault circular representation for contact (right). The nodal contact area assigned to the beam nodes is represented by an encompassing circular cross-section equal to the encompassing radius times the sum of the adjacent beam element lengths projected onto a plane normal to the nodal contact force vector.

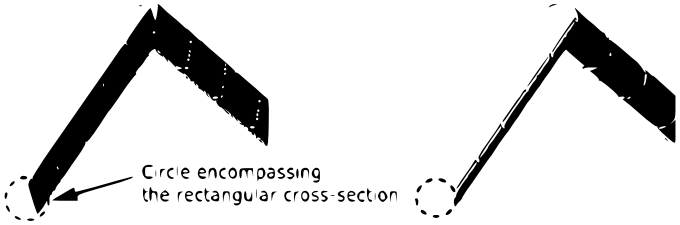


Figure 6: Beam element profile rendering for circular versus rectangular cross-section beams.

Each nodal value of the effective radius corresponds to the minimum encompassing radius among adjacent beam or truss elements. Effective values of nodal radii are interpolated within a beam or truss element for the purposes of the contact calculations.

Input File Usage: By default, contact calculations for noncircular beams are based on the internally generated surface-element representation of their cross-sections. For circular beams and pipes, the contact calculations use contact edges and nodes with a radius equal to the section radius.

Use the following option to activate contact calculations based on the internally generated surface-element representation of the beam cross-section for the entire model:

**CONTACT CONTROLS ASSIGNMENT*, BEAM CROSS SECTION=EXACT

Use the following option to activate contact calculations based on the circumscribed circle representation of the beam cross-section for the entire model:

**CONTACT CONTROLS ASSIGNMENT*, BEAM CROSS SECTION=CIRCUMSCRIBED CIRCLE

Abaqus/CAE Usage: Interaction module: Create interaction: General contact: Contact Controls Assignment:

Beam Cross Section assignment: Select Circumscribed Circle or Exact.

Contact Surface Representation for Shells

The default contact surface representation for a part modeled with shell elements is a double-sided shell reference surface with consideration of shell thickness and shell offset. *Figure 7* shows the model with and without shell rendering. In this example, the perimeter of one part is in contact with the shell element faces of the other part.

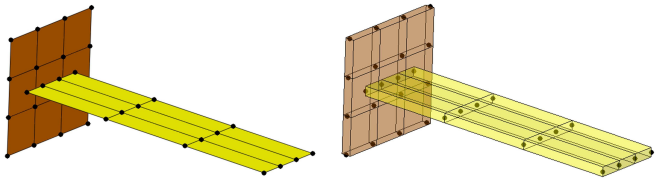


Figure 7: Two bodies meshed with shell elements in contact, with and without thickness rendering.

Optionally, you can specify that Abaqus/Standard internally generate surface elements near shell perimeters, as shown in *Figure 8*. Surface elements are created along the perimeter with normal directions corresponding to the perimeter direction. These elements represent the shell perimeter area. A row of surface elements is added on the top side and on the bottom side of the shell adjacent to the perimeter. These elements share nodes with

perimeter surface elements and interior shell elements. The contact thickness of nodes shared by surface elements and interior shell elements is half the shell thickness. Internally generated surface element nodes along the perimeter have zero contact thickness. The contact thickness of slanted surface elements is interpolated from nodal values, which provides smooth transitions for contact sliding across faces. You can remove these surface elements from the postprocessor display to return to a view such as that shown on the left in *Figure 7*.

Internally generating faces along shell perimeters offers the following advantages:

- The surface-to-surface contact formulation can be used to model contact involving shell perimeters, resulting in a more detailed resolution of interface interactions;
- Cohesive contact can resist relative rotation for contact involving shell perimeters.

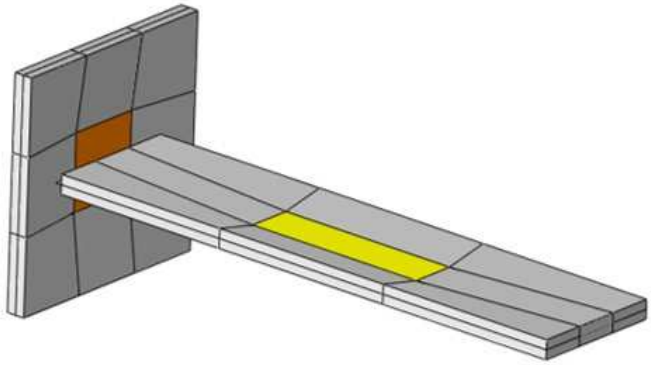


Figure 8: Surface configuration without thickness rendering when you introduce surface elements near shell perimeters automatically.

Figure 9 compares vertical displacement results for a moment load at the tip of the cantilever for simulations with:

- A bond formed as a surface-based tie constraint for a beam meshed with shell elements acting as a secondary surface to a wall meshed with shells;
- A bond formed as a permanent cohesive bond with default cohesive stiffness for a beam meshed with shell elements (and internally generated surface elements) and a wall meshed with shell elements;
- A bond formed as a permanent cohesive bond with default cohesive stiffness for a beam meshed with shell elements (and internally generated surface elements) and a wall meshed with solid elements.

Tip displacements vary by 2% across the simulations shown.

Surface-based tie constraints constrain rotational degrees of freedom only if both surfaces have rotational degrees of freedom. *Table 1* summarizes which cases of bonded interfaces involving a shell perimeter are able to resist relative rotation for a bond.

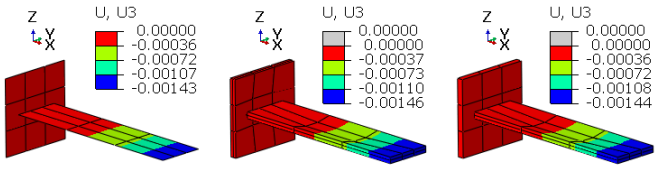


Figure 9: Comparison of tip deflection for three cantilever models.

Table 1: Summary of which cases of bonded interfaces involving a shell perimeter will resist relative rotation.

Bond Type	Internally generated faces at shell perimeter	Beam mesh	Wall mesh	Resist relative rotation at interface
Surface-based tie	No	Shell	Shell	Yes✔
			Solid	No✘
Cohesive behavior in general contact	No		Shell	No✘
			Solid	
	Yes		Shell	Yes✔
			Solid	

The motion of each surface element node is driven by the corresponding shell perimeter node via an internally generated beam MPC (see *Using MPC Type BEAM*). Abaqus/Standard redistributes the contact forces acting on these nodes back to the original shell nodes as forces and moments, ensuring the correct shell response. The surface element-based surfaces inherit the general contact property assigned to the corresponding shell surfaces. Meshes of the surface elements along shell perimeters are available for postprocessing (see *Output for Beams with Actual Beam Cross-Section Representation and Shells with Actual Shell Perimeter Representation*).

Input File Usage: Use the following option to activate contact calculations based on the shell perimeter edges (default):

**CONTACT CONTROLS ASSIGNMENT*, SHELL REPRESENTATION=REFERENCE SURFACE

Use the following option to activate contact calculations based on the surface element representation of the shell perimeters:

**CONTACT CONTROLS ASSIGNMENT*, SHELL REPRESENTATION=EXTRA PERIMETER FACES

Abaqus/CAE Usage: Defining the shell representation assignment is not supported in Abaqus/CAE.

Contact Surface Representation for Threaded Interfaces

Introduction

Accurate modeling of bolted connections depends on accounting for thread geometry especially when thread surfaces experience large relative motions during bolt tightening and loosening. Meshing the exterior or interior threads of bolts and nuts with solid elements directly (along with the rest of the part) might be inefficient for the following reasons:

- mesh generation can be cumbersome to capture the thread geometry accurately and
- the refined meshes needed are computationally expensive.

You can now provide a nominal (usually coarse) solid mesh with a cylindrical geometry, as shown in *Figure 10*, for the bolt and a corresponding one for the mating part.

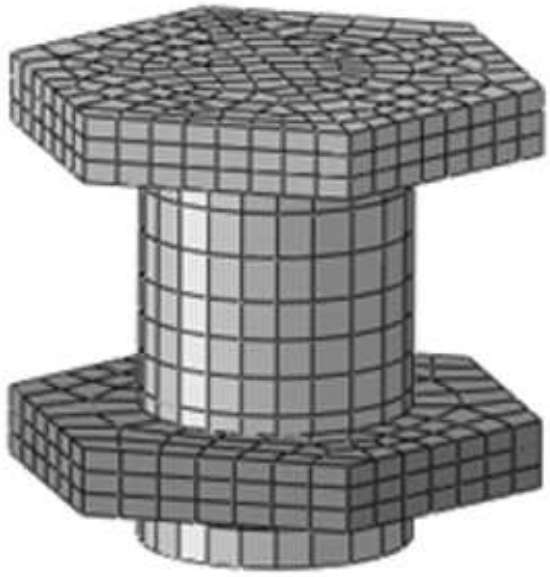


Figure 10: Nominal meshes of a bolt and a nut with cylindrical surfaces instead of threads.

You can then specify general contact to use reference input thread geometry parameters (see *Figure 11*) to generate separate surface meshes for the threads (see *Figure 12*) on the bolt and the nut automatically.

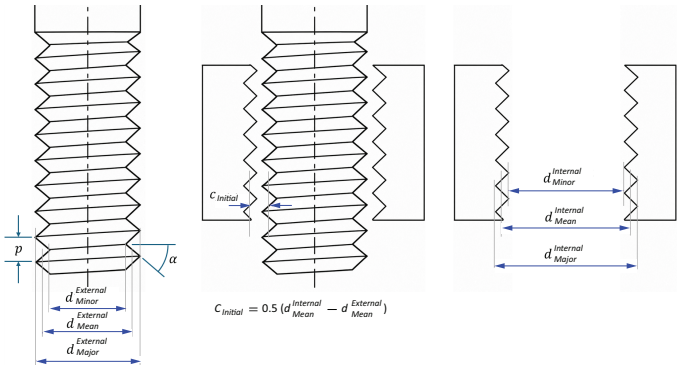


Figure 11: V-shaped thread geometry.

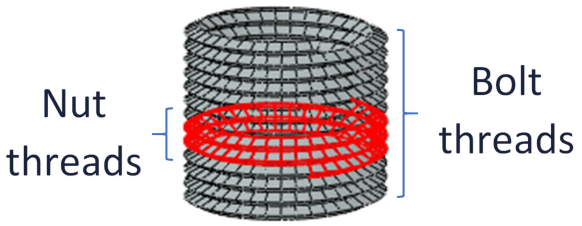


Figure 12: Automatically generated thread meshes of a bolt and nut.

Thread surface meshes allow the thread geometry to be meshed independently of the underlying solid part, reducing meshing complexity and computational effort while maintaining accurate contact behavior under large relative sliding.

Abaqus/Standard automatically constrains the surface meshes to their corresponding nominal solid meshes, which transfers the contact forces generated on these thread surfaces to the underlying nominal meshes, as shown in *Figure 13*. The thread surface meshes inherit the interaction properties assigned to their corresponding nominal solid mesh surfaces.

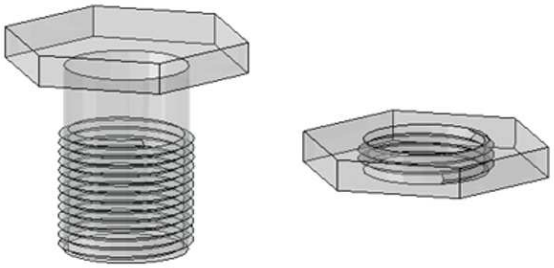


Figure 13: Automatically generated thread meshes connected to respective nominal bolt and nut meshes.

Figure 14 shows bolt tightening where the animation on the left shows the two layers being pinched together as the bolt is tightened into the nut. The animation on the right shows contact pressure developing on the thread surface for the bolt (for model details, see *Contact surface representation of threads*).

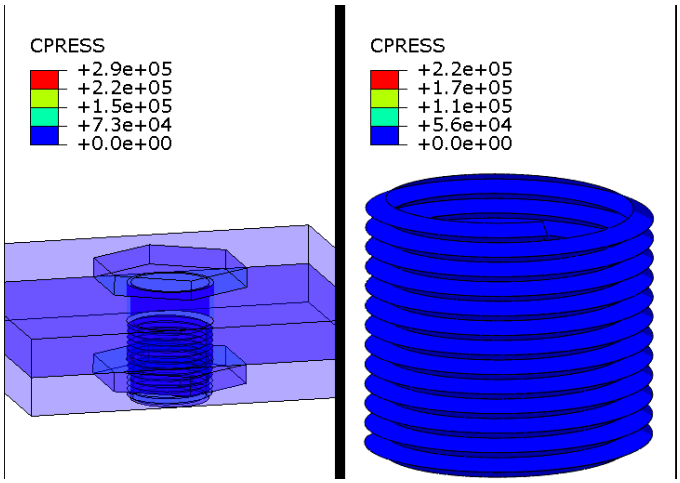


Figure 14: Bolt tightening example.

Bolt tightening and loosening involves large relative sliding between thread surfaces. By default, general contact uses the finite-sliding tracking approach, which accurately captures the large relative sliding of the surface meshes representing the threads.

For approximate thread modeling, Abaqus/Standard also provides a small-sliding tracking approach (see *Generating Contact Normal Directions Based on a Reference Thread Geometry*) that uses a planar approximation of the implied thread geometry based on reference thread parameters. With this approach you need to provide only the nominal meshes. However, Abaqus/Standard does not generate surface elements for the threads. The small-sliding approach limits the capability to model small relative motions. Results can become inaccurate in bolt loosening and tightening applications that involve large relative motions between the thread surfaces compared to the radius of curvature of the threads.

Automatically Generated Surface Element Representations of Threads

You can specify the thread geometry parameters along with two nominal surfaces that represent the outer surface of the externally threaded bolt (or bolt-like) part and the inner surface of the internally threaded nut (or nut-like)

part. Abaqus/Standard supports only underlying solid meshes for the nominal surfaces, and the nominal surfaces must be cylindrical.

Input File Usage: Use the following options to activate the surface element representation of the external and internal threads:

**CONTACT CONTROLS ASSIGNMENT*, THREAD MESH GENERATION
surface_name_external, surface_name_internal, thread_interface_name

**THREADED INTERFACE*, NAME=thread_interface_name
half-thread angle, pitch

Abaqus/CAE Usage: Automatically generating thread meshes is not supported in Abaqus/CAE.

Thread Handedness

You can define right-handed or left-handed threads.

Input File Usage: Use the following option to define right-handed threads (default):

**THREADED INTERFACE*, NAME=thread_interface_name,
HANDEDNESS=RIGHT

Use the following option to define left-handed threads:

**THREADED INTERFACE*, NAME=thread_interface_name, HANDEDNESS=LEFT

Abaqus/CAE Usage: Automatically generating thread meshes is not supported in Abaqus/CAE.

Thread Geometric Parameters

Abaqus/Standard supports a symmetric V-shape thread geometry, as shown in *Figure 11*. You specify the symmetric half-thread angle, α, and the pitch, p. The half-thread angle is assumed to be 30° by default. The threads can be either left-handed or right-handed.

Optionally, you can specify the initial radial clearance distance and the external thread diameter, which are interdependent. You can provide an external diameter, $d_{Spec}^{External}$, on the bolt (or bolt-like) part. Then, you can specify if this diameter should be interpreted as a mean diameter (default) or, alternatively, as a major or a minor diameter. If you do not specify the external diameter, Abaqus/Standard calculates it automatically from the nominal cylindrical mesh of the bolt and interprets it based on user specification.

Table 2 shows how Abaqus/Standard determines the mean diameter, of the external threads ($d_{Mean}^{External}$), depending on whether you provided the value of the external thread diameter ($d_{Spec}^{External}$) or allowed Abaqus/Standard to calculate it automatically.

Table 2: Mean diameter of the external threads.

External thread diameter specification	Interpreted value of the external thread diameter	Mean diameter of external threads, $d_{Mean}^{External}$
User-specified external thread diameter ($d_{Spec}^{External}$)	Mean diameter (default)	$d_{Spec}^{External}$
	Minor diameter	$d_{Spec}^{External} + \frac{p}{2\tan(\alpha)}$
	Major diameter	$d_{Spec}^{External} - \frac{p}{2\tan(\alpha)}$
External thread diameter derived from the nominal mesh diameter $d_{No\,min\,al}^{External}$	Mean diameter (default)	$d_{No\,min\,al}^{External}$
	Minor diameter	$d_{No\,min\,al}^{External} + \frac{p}{2\tan(\alpha)}$
	Major diameter	$d_{No\,min\,al}^{External} - \frac{p}{2\tan(\alpha)}$

Abaqus/Standard calculates the mean diameter of the internal threads, $d_{Mean}^{Internal}$, used to generate the surface mesh for internal threads based on the specified clearance, $c_{Initial}$, according to the following formula:

$$d_{Mean}^{Internal} = d_{Mean}^{External} + 2c_{Initial}.$$

If you do not specify the initial radial clearance distance, Abaqus/Standard determines the mean diameter, $d_{Mean}^{Internal}$, of the internal threads as shown in *Table 3*.

Table 3: Mean diameter of the internal threads if you do not specify the radial clearance distance.

Interpreted value of the internal thread diameter	Mean diameter of internal threads, $d_{Mean}^{Internal}$
Mean diameter (default)	$d_{No\,min\,al}^{Internal}$
Minor diameter	$d_{No\,min\,al}^{Internal} + \frac{p}{2\tan(\alpha)}$
Major diameter	$d_{No\,min\,al}^{Internal} - \frac{p}{2\tan(\alpha)}$

The nominal mesh diameter of the internal threads is denoted by $d_{No\,min\,al}^{Internal}$. In this case, the initial radial clearance distance, $c_{Initial}$, is recovered for the contact calculations as

$$c_{Initial} = \frac{1}{2}\left(d_{Mean}^{Internal} - d_{Mean}^{External}\right).$$

Input File Usage: Use the following option to specify the clearance and/or the diameters of the external and internal threads:

**THREADED INTERFACE*, NAME=thread_interface_name
half-thread angle, pitch
initial radial clearance distance, diameter of the external threads, MEAN/MINOR/MAJOR for external threads, MEAN/MINOR/MAJOR for internal threads

Abaqus/CAE Usage: Automatically generating thread meshes is not supported in Abaqus/CAE.

Thread Extents and Orientation of Generated Thread Meshes

Based on the user-supplied nominal surfaces of the threaded parts, Abaqus/Standard can automatically calculate the external thread extent, the internal thread extent, and the axis of the threads. Alternatively, you can specify these values explicitly.

Providing Points for User-Specified Extent and Axis Definitions

To specify the thread extents and/or the axis of the threads explicitly, you must provide points for the external threads (PE1, PE2) and/or points for the internal threads (PI1, PI2), as shown in *Figure 15*).

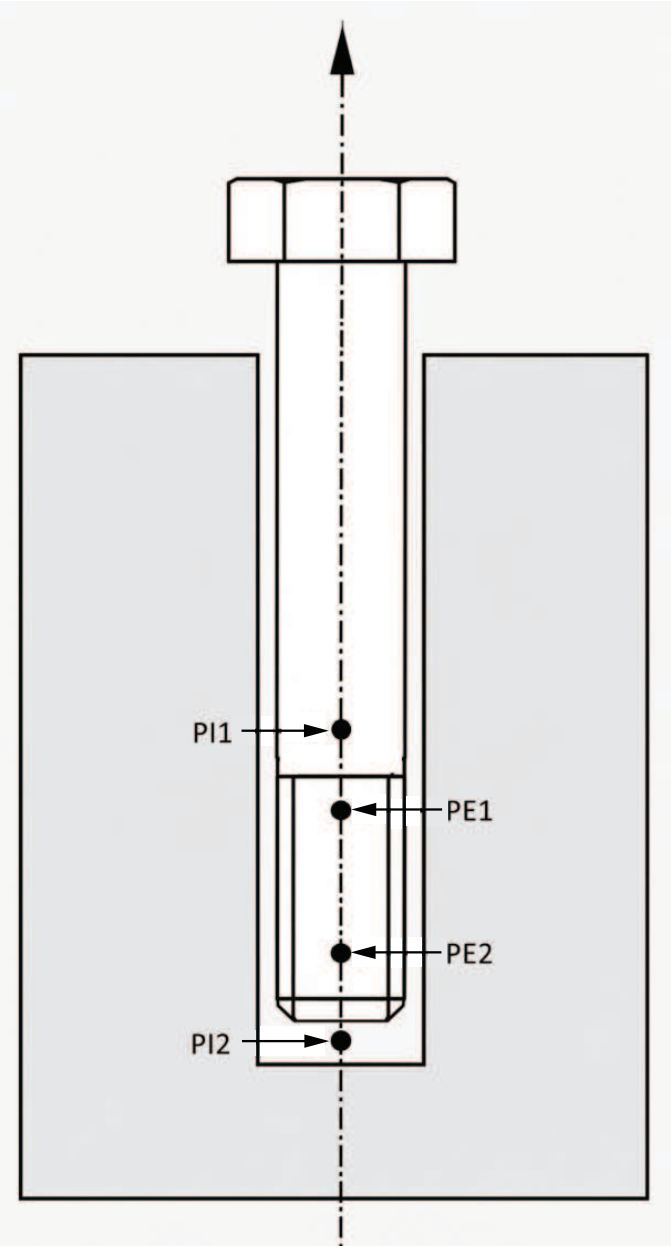


Figure 15: User-specified external thread points (PE1, PE2) and internal thread points (PI1, PI2) to define the thread extents and/or the axis of the threads.

To specify the points, you can either provide the coordinates of the points or you can specify two nodes.

Input File Usage: Use the following option to specify coordinates to control the axial range over which threads are generated and/or to specify the threads axis:

**THREADED INTERFACE*, NAME=thread_interface_name, POINTS=COORDINATES

Use the following option to provide nodes to control the axial range over which threads are generated and/or to specify the threads axis:

**THREADED INTERFACE*, NAME=thread_interface_name, POINTS=NODES

Abaqus/CAE Usage: Automatically generating thread meshes is not supported in Abaqus/CAE.

Axis Definition

You can allow Abaqus/Standard to determine the axis, or you can define the axis directly.

Input File Usage: Use the following option to adopt the automatically calculated axis derived from the nominal surfaces of the externally and the internally threaded parts:

**THREADED INTERFACE*, NAME=thread_interface_name, AXIS=AUTOMATIC

Use the following option to adopt the axis specified by the user-provided external thread points:

**THREADED INTERFACE*, NAME=thread_interface_name, AXIS=EXTERNAL THREAD POINTS

Use the following option to adopt the axis specified by the user-provided internal thread points:

**THREADED INTERFACE*, NAME=thread_interface_name, AXIS=INTERNAL THREAD POINTS

Abaqus/CAE Usage: Automatically generating thread meshes is not supported in Abaqus/CAE.

External Thread Extent Definition

You can allow Abaqus/Standard to calculate the external thread extent, or you can specify the external thread extent directly.

Input File Usage: Use the following option to adopt the external thread extent automatically calculated from the nominal surface of the externally threaded part:

**THREADED INTERFACE*, NAME=thread_interface_name, EXTERNAL THREAD EXTENT=AUTOMATIC

Use the following option to adopt the external thread extent specified by the user-provided points for the external threads:

**THREADED INTERFACE*, NAME=thread_interface_name, EXTERNAL THREAD EXTENT=SPECIFY POINTS

Abaqus/CAE Usage: Automatically generating thread meshes is not supported in Abaqus/CAE.

Internal Thread Extent Definition

You can allow Abaqus/Standard to calculate the internal thread extent, or you can specify the internal thread extent directly.

Input File Usage: Use the following option to adopt the internal thread extent automatically calculated from the nominal surface of the internally threaded part:

**THREADED INTERFACE*, NAME=thread_interface_name, INTERNAL THREAD EXTENT=AUTOMATIC

Use the following option to adopt the internal thread extent specified by the user-provided points for the internal threads:

**THREADED INTERFACE*, NAME=thread_interface_name, INTERNAL THREAD EXTENT=SPECIFY POINTS

Abaqus/CAE Usage: Automatically generating thread meshes is not supported in Abaqus/CAE.

Surface Mesh Control

You can use first-order surface elements or second-order surface elements (default) for the automatically generated thread meshes. The mesh size for the automatically generated surface elements is chosen to be comparable to the average mesh size of the nominal surfaces of the threaded parts. Abaqus/Standard generates at least one surface element over every half pitch of the thread.

Input File Usage: Use the following option to use second-order surface elements for the automatically generated thread meshes (default):

**THREADED INTERFACE*, NAME=thread_interface_name, FACETS=QUADRATIC

Use the following option to use first-order surface elements for the automatically generated thread meshes:

**THREADED INTERFACE*, NAME=thread_interface_name, FACETS=LINEAR

Abaqus/CAE Usage: Automatically generating thread meshes is not supported in Abaqus/CAE.

Types of Contact Formulations within General Contact Targeting Various Scenarios

The general contact algorithm in Abaqus/Standard offers capabilities to model surface-to-surface contact, edge-to-surface contact, edge-to-edge contact, and vertex-to-surface contact with the default finite-sliding tracking approach. The surface-to-surface contact formulation is the primary formulation with three-dimensional models and is the only formulation for general contact with two-dimensional and axisymmetric models.

You can specify the small-sliding tracking approach to model surface-to-surface contact, but small sliding is not supported with edges and vertices (see *Specifying Small Sliding within General Contact*). The finite-sliding tracking approach is the preferred approach due to its generality in handling arbitrary relative surface motions. However, small sliding in general contact provides a highly automated and convenient contact modeling approach in situations where relative surface motions are known to be small.

Pure heat transfer and coupled thermal-electrical contact interactions support surface-to-surface contact as well as edge-to-surface contact with a beam edge acting as the main contact entity.

For three-dimensional models, the surface-to-surface contact formulation primarily treats cases with contact over an area of dimensions significant compared to the surface facet dimensions, such as the case on the left in *Figure 16*. General contact with finite sliding for three-dimensional models uses the other contact formulations as supplementary formulations. For example, the second, third, and fourth cases in *Figure 16* is treated with the edge-to-surface, edge-to-edge, and vertex-to-surface formulations, respectively. General contact also uses the supplementary formulations to treat contact involving three-dimensional beam and truss elements. The surface-to-surface, edge-to-surface, and vertex-to-surface formulations can treat contact interactions in which one surface (which does not represent an edge or a vertex) is an analytical rigid surface. General contact does not consider contact between two analytical rigid surfaces.

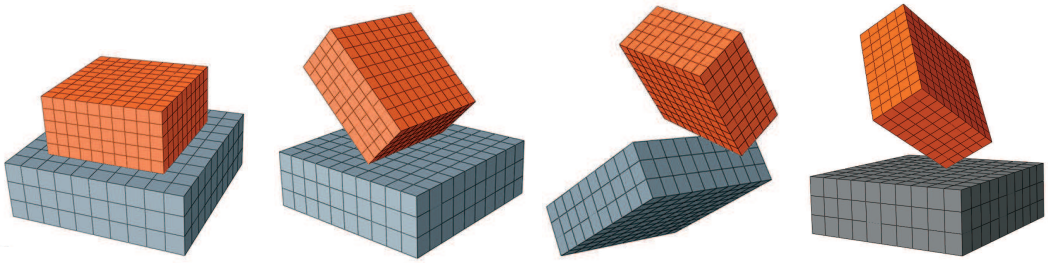


Figure 16: Contact scenario types for two blocks.

Transitions between the predominant type of contact formulation active in a local region are common. For example, the edge-to-surface contact formulation would be predominant at the stage of the snap-fit simulation shown in *Figure 17* because the active contact zone corresponds to a feature edge.

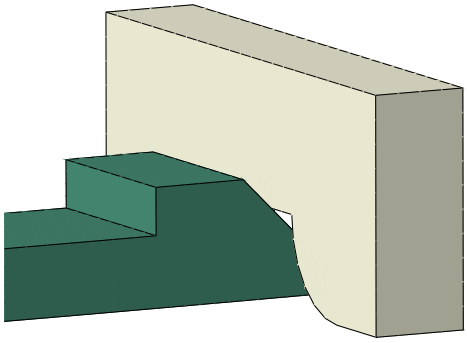


Figure 17: Snap-fit example involving feature edge-to-surface contact with an oblique angle between surface normals in the contact region.

Upon further insertion, the surface-to-surface contact formulation would become predominant once the top surface of the darker colored part is in contact with the other part over a significant area. General contact automatically handles transitions between predominant contact formulations as contact conditions evolve. Multiple types of contact constraints are locally active during transitions. The supplementary contact formulations are always enforced with a penalty method, which helps avoid numerical issues with “over-constraints” while multiple constraint types are active.

For two-dimensional and axisymmetric models, surfaces based on solid, shell, beam, and truss elements consist of linear or quadratic segments (sometimes referred to as two-dimensional faces). For three-dimensional models, beam and truss surfaces are comprised of segments (sometimes referred to as "edges"), whereas surfaces based on three-dimensional solid, shell, and membrane elements consist of triangular or quadrilateral faces (and edges of face boundaries). Due to the similarity of two-dimensional surfaces among element types, the two-dimensional surface-to-surface contact formulation can treat surfaces based on solid, shell, beam, and truss elements. The

three-dimensional surface-to-surface contact formulation treats surfaces based on solid and shell elements but does not treat beam or truss surfaces.

See *Surface-to-Surface Contact Discretization* for additional discussion of the surface-to-surface contact formulation. See *Edge-to-Surface Contact Scenarios*, *Edge-to-Edge Contact Scenarios*, and *Vertex-to-Surface Contact Scenarios* below, for more information on the supplementary contact formulations.

Including General Contact in an Analysis

You can specify an overall general contact definition in multiple portions as model and step data in Abaqus/Standard. However, you must define the envelope of contact interactions eligible to be considered by general contact in one or more of these portions as model data.

Input File Usage: Use the following option to indicate the beginning of a portion of the general contact specification:

**CONTACT*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Step: Initial**, **General contact (Standard)**

Defining the General Contact Domain

The general contact domain is the envelope of potential interactions for a simulation and the associated surface entities (nodes, faces, etc.). You determine the general contact domain based on general contact inclusions and exclusions that you specify at the model level and implied contact exclusions based on other model-level features such as contact pairs and surface-based tie constraints. If you specify an all-inclusive contact inclusion at the model level together with a contact exclusion between nonoverlapping surfaces A and B, the general contact domain includes potential interactions between all surface entities of the model (even for surfaces that you have not explicitly defined and named) except for potential interactions between surfaces A and B. Entities of surfaces A and B can still experience self-contact and contact with surfaces other than A and B. Exclusions take precedence over inclusions at the model level regardless of which definition appears first.

Named surfaces used in the definition of general contact can span multiple unattached bodies. For example, if you allow a named surface spanning multiple bodies to possibly contact itself, contact will be enforced between the entities of that surface from the same body or different bodies.

You can deactivate interactions that you specify at the model level and subsequently reactivate them across steps. Define contact exclusions at the step level to deactivate certain interactions of the contact domain. Define contact inclusions to reactivate the interactions in a subsequent step (see *Deactivating and Reactivating Contact Interactions across Steps*).

Specifying Contact Inclusions

Define contact inclusions to specify the regions of the model that Abaqus/Standard should consider for contact purposes.

Specifying “Automatic” Contact for the Entire Model

You can specify self-contact for a default unnamed, all-inclusive surface defined automatically by Abaqus/Standard. This default surface contains, with the exceptions noted below, all exterior element faces and all analytical rigid surfaces. This is the simplest way to define the contact domain.

The default surface does not include faces that belong only to cohesive elements. In fact, the default surface is generated as if cohesive elements were not present. See *Modeling with Cohesive Elements* for further discussion of contact modeling issues related to cohesive elements.

Input File Usage: Use both of the following options to specify "automatic" contact for the entire model:

**CONTACT*
**CONTACT INCLUSIONS*, ALL EXTERIOR

The **CONTACT INCLUSIONS* option should have no data lines when the ALL EXTERIOR parameter is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Included surface pairs: All* with self**

Specifying Individual Contact Interactions

Alternatively, you can define the general contact domain directly by specifying the individual contact surface pairings. Self-contact is modeled only if the two surfaces specified in a pair overlap (or are identical) and are modeled only in the overlapping region. In some cases you can improve computational performance and robustness by including only portions of surfaces in the general contact domain that will experience contact during an analysis.

You can include multiple surface pairings in the contact domain. All of the surfaces specified must be element-based surfaces. Edge-based surfaces cannot be included using this method.

Input File Usage: Use both of the following options to specify individual contact interactions:

**CONTACT*
**CONTACT INCLUSIONS*
surface_1, *surface_2*

At least one data line must be specified when the ALL EXTERIOR parameter is omitted. You can leave either or both of the data line entries blank, but each data line must contain at least a comma; an error message will be issued for empty data lines. If you omit the first surface name, Abaqus/Standard assumes the default unnamed, all-inclusive, automatically generated surface. If the second surface name is omitted or is the same as the first surface name, contact between the first surface and itself is assumed. Leaving both data line entries blank is equivalent to using the ALL EXTERIOR parameter.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Included surface pairs: Selected surface pairs: Edit**, select the surfaces in the columns on the left, and click the arrows in the middle to transfer them to the list of included pairs

Examples

The following input specifies that contact should be enforced between the default all-inclusive, automatically generated surface and *surface_2*, including self-contact in any overlap regions:

```
*CONTACT
*CONTACT INCLUSIONS
 , surface_2
```

Either of the following methods can be used to define self-contact for *surface_1*:

```
*CONTACT
*CONTACT INCLUSIONS
surface_1,
```

or

```
*CONTACT
*CONTACT INCLUSIONS
surface_1, surface_1
```

Specifying Contact Exclusions

You can refine the contact domain definition by specifying the regions of the model to exclude from contact. Possible motivations for specifying contact exclusions include:

- avoiding physically unreasonable contact interactions;
- improving computational performance by excluding parts of the model that are not likely to interact.

Contact is ignored for all the surface pairings specified, even if these interactions are specified directly or indirectly in the contact inclusions definition.

Multiple surface pairings can be excluded from the contact domain. All the surfaces specified must be element-based surfaces. Keep in mind that surfaces can be defined to span multiple unattached bodies, so self-contact exclusions are not limited to exclusions of single-body contact.

Input File Usage: Use both of the following options to specify contact exclusions:

**CONTACT*
**CONTACT EXCLUSIONS*
surface_1, *surface_2*

You can leave either or both of the data line entries blank. If you omit the first surface name, Abaqus/Standard assumes the default unnamed, all-inclusive, automatically generated surface. If you omit the second surface name or you use the same as the first surface name, contact between the first surface and itself is excluded from the contact domain.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Excluded surface pairs: Edit**, select the surfaces in the columns on the left, and click the arrows in the middle to transfer them to the list of excluded pairs

Automatically Generated Contact Exclusions

Abaqus/Standard automatically generates contact exclusions for general contact in some situations.

- Contact exclusions are generated automatically for interactions that are defined with the contact pair algorithm or surface-based tie constraints to avoid redundant (and possibly inconsistent) enforcement of these interaction constraints. For example, if a contact pair is defined for `surface_1` and `surface_2` and "automatic" general contact is defined for the entire model, Abaqus/Standard generates a contact exclusion for general contact between `surface_1` and `surface_2` so that interactions between these surfaces are modeled only with the contact pair algorithm. These automatically generated contact exclusions are in effect throughout the analysis.
- Abaqus/Standard automatically generates contact exclusions for self-contact of each rigid body in the model, because a rigid body cannot contact itself.

- When you specify pure main-secondary contact surface weighting for a particular general contact surface pair, contact exclusions are generated automatically for the main-secondary orientation opposite to that specified (see *Contact Controls Specific to General Contact in Abaqus/Standard* for more information on this type of contact exclusion).
- Abaqus/Standard assigns default pure main-secondary roles for contact involving disconnected bodies within the general contact domain, and contact exclusions are generated by default for the opposite main-secondary orientations. Options to override the default pure main-secondary assignments with alternative pure main-secondary assignments or balanced main-secondary assignments are discussed in *Contact Controls Specific to General Contact in Abaqus/Standard*.
- Contact exclusions are generated automatically for portions of surfaces that are severely overclosed in the initial configuration of the model. See *Contact Initialization for General Contact in Abaqus/Standard* for more information.

Example: Automatically Generated Contact Exclusions

The following input specifies that the contact domain is based on self-contact of an all-inclusive, automatically generated surface but that contact (including self-contact in any overlap regions) should be ignored between the all-inclusive, automatically generated surface and *surface_2*:

```
*CONTACT
*CONTACT INCLUSIONS, ALL EXTERIOR
*CONTACT EXCLUSIONS
 , surface_2
```

Use either of the following methods to exclude self-contact for *surface_1* from the contact domain:

```
*CONTACT EXCLUSIONS
surface_1,
```

or

```
*CONTACT EXCLUSIONS
surface_1, surface_1
```

Deactivating and Reactivating Contact Interactions across Steps

You can deactivate general contact interactions established at the model level and subsequently reactivate them across steps using contact exclusions and contact inclusions, respectively. Abaqus/Standard issues an error message if you attempt to activate an interaction in a step that is not eligible for contact according to the model-level general contact definition. The model-level specification establishes the envelope of eligible contact interactions.

Consider the following example: A particular forming tool is relevant only to contact starting in Step 3 of a simulation. For convenience, set the original position of this tool where it will eventually be introduced in the simulation. Exclude this part from contact interactions for the first two steps to avoid undesired interference. You will:

- Include the interactions involving this tool at the model level.
- Specify a contact exclusion for this part with the entire model in Step 1 to deactivate these contact interactions.
- Specify a contact inclusion for this part with the entire (or desired subset) of the model in Step 3 to reactivate these contact interactions.
- Specify any needed contact initialization instructions for the newly reactivated contact interactions in Step 3.

Contact deactivation and subsequent reactivation in analysis steps do not require use of the same surface names (nor the same extents of the surfaces) that are used to define the model-level contact inclusions that establish the envelope of eligible contact interactions.

If contact deactivation occurs for contact interactions that were actively in contact at the end of the previous step, contact forces (and heat or electrical fluxes in the case of additional thermal and electrical interactions) are ramped down for contact constraints that were active at the end of the previous general step.

Any contact state associated with friction, cohesive contact, etc., is initialized on contact reactivation for interactions that were previously deactivated. Specifying a contact inclusion in a step has no effect on the contact state for contacts that were not previously deactivated.

Some steps might involve deactivation of some contact interactions and reactivation of other contact interactions. In the case of overlapping interactions between contact inclusions and contact exclusions specified within a step, the contact exclusions take precedence.

Step-dependent activation and deactivation are not supported for heat transfer, coupled thermal-electrical, and coupled thermal-electrochemical procedures. Similarly, step-dependent activation and deactivation are not allowed for general contact with XFEM surfaces anywhere in the model. For the small-sliding tracking approach in general contact, portions of the model where small-sliding is specified also cannot participate in step-dependent activation and deactivation.

Example: Deactivating and Reactivating Contact Interactions across Steps

In this example, the contact domain established at the model level includes all potential contact interactions in a model using the following options.

```
*CONTACT
*CONTACT INCLUSIONS, ALL EXTERIOR
```

Contact interactions between *surf_A* and *surf_B* are deactivated in Step 2 using the following options.

```
*STEP
…
*CONTACT
*CONTACT EXCLUSIONS
surf_A, surf_B
…
*END STEP
```

Contact interactions between an implied all-inclusive surface and *surf_Reactivate* are reactivated, and all interactions involving *surf_C* are deactivated in Step 3 using the following options (here *surf_Reactivate* is a subset of *surf_A*):

```
*STEP
…
*CONTACT
*CONTACT INCLUSIONS
, surf_Reactivate
*CONTACT EXCLUSIONS
, surf_C
…
*END STEP
```

Edge-to-Surface Contact Scenarios

The general contact algorithm can consider three-dimensional edge-to-surface contact. In addition to modeling contact between segments of beam or truss elements and faceted surfaces, it is more effective at resolving some interactions than the surface-to-surface contact formulation. *Figure 17* and *Figure 18* show examples in which the edge-to-surface contact formulation is most effective for resolving contact.

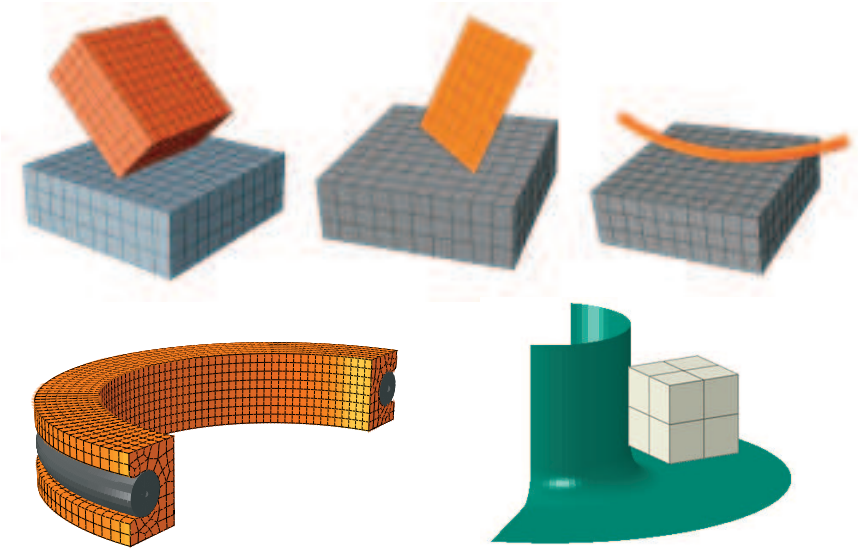


Figure 18: Edge-to-surface contact examples.

The bottom-right example in *Figure 18* shows a feature edge of an element-based surface in contact with an analytical rigid surface. The feature edges of an analytical rigid surface do not act as edges in the edge-to-surface formulation. You can enhance convergence behavior by avoiding feature edges and corners in analytical rigid surfaces used for contact: Instead, use smooth analytical rigid surfaces with continuously varying surface normal directions for contact.

Contact edges representing three-dimensional beam and truss elements have a circular cross-section (regardless of the actual cross-section of the beam or truss element), unless you enable actual beam cross-section representation for beams with noncircular cross-sections (see *Contact Surface Representation for Beams*). The radius of a contact edge representing a three-dimensional truss element is derived from the cross-sectional area specified on the truss section definition (it is equal to the radius of a solid circular section with an equivalent cross-sectional area). For three-dimensional beams with circular cross-sections, the radius of the contact edge is equivalent to the section radius. For three-dimensional beams with noncircular cross-sections, the radius of the contact edge is equal to the radius of a circumscribed circle around the section. Edge-to-surface contact for three-dimensional beam or truss elements is activated by including the associated surfaces into the general contact domain. By default, the all-inclusive surface contains surfaces based on beam or truss elements.

The surface thickness for two-dimensional beam segments corresponds to the in-plane thickness of the beam section. The surface thickness for two-dimensional truss segments corresponds to the radius of a solid circular section with an equivalent cross-sectional area as specified for the truss section. Beams and trusses are treated only with the surface-to-surface contact formulation in general contact, as discussed in *Types of Contact Formulations within General Contact Targeting Various Scenarios*.

Beam section offsets, specified either as part of the beam cross-section definition or directly, are neglected for contact interactions in Abaqus/Standard.

The edge-to-surface contact formulation is commonly used to resist penetrations of feature edges of one surface into a relatively smooth portion of another surface (which might be an analytical rigid surface), with the contact normal direction based on the relatively smooth surface. The main and secondary roles of surfaces in the edge-to-surface contact formulation are reversed for some situations involving large-diameter beams. By default, if half of the beam radius exceeds the facet dimensions of the other surface, the beam acts as the main surface such that the edge-to-surface contact formulation resists penetrations of a smooth portion of a neighboring solid or shell surface into a beam, with the contact normal direction based on the radial direction of the beam. The bottom-left example in *Figure 18* corresponds to such a case with a relatively large diameter beam (see *Main*

and Secondary Surface Roles of a Contact Formulation for details about how to control main and secondary assignment).

In pure heat transfer and coupled thermal-electrical analyses, only the edge-to-surface formulation with the one-dimensional line ("beam") element–based surfaces acting as the main is supported. Consequently, these line element–based surfaces act as the main by default. The line elements are intended to model higher dimensional physics with a one-dimensional idealization. They are expected to have a cross-sectional (beam) radius that is comparable to the facet dimensions of the surrounding surfaces to obtain accurate results for thermal/electrical interactions.

The edge-to-surface contact formulation considers twisting of beams only for cases in which the contact normal is based on the radial direction of the beam. When considered, beam twisting influences the calculation of incremental slip.

By default, when a surface is used in a general contact interaction, all applicable facets are included in the contact definition along with edges of solid and shell elements with feature angles of at least 45°. See *Feature Edges* for a discussion of controls related to which feature edges are considered for edge-to-surface contact. Edge-to-surface contact constraints never participate in thermal, electrical, or pore pressure contact properties. For example, in a coupled temperature-displacement analysis, surface-to-surface constraints can influence mechanical and thermal interactions; but, if edge-to-surface constraints are included, they will only help resist penetrations.

The contact area associated with a feature edge depends on the mesh size; therefore, contact pressures (in units of force per area) associated with edge-to-surface contact are mesh dependent.

Edge-to-Edge Contact Scenarios

The general contact algorithm can optionally consider three-dimensional edge-to-edge contact except on crack surfaces for enriched elements. Feature edges on solid and shell-like surfaces, shell perimeter edges, and edges representing beams (and trusses) can be included. *Figure 19* shows examples in which the edge-to-edge contact formulation is most effective for resolving contact.

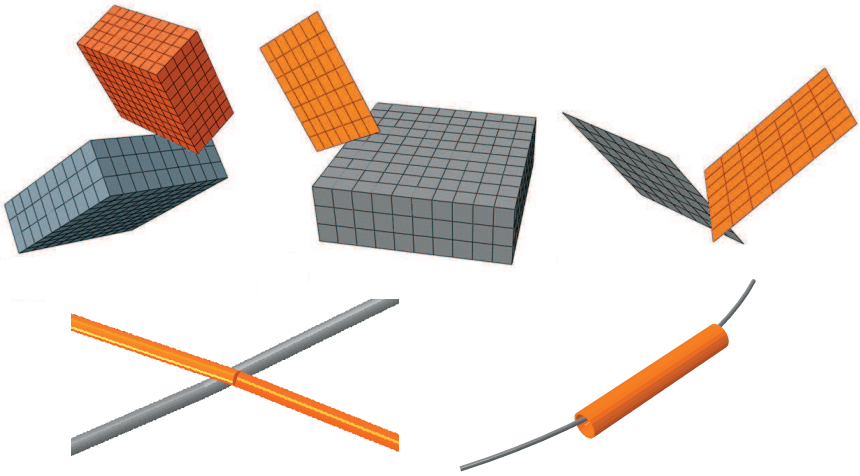


Figure 19: Edge-to-edge contact examples.

Two edge-to-edge contact formulations are available. One formulation bases the contact normal direction on the cross product between the two respective edges considered for contact, and the other formulation uses a radial direction of one of the beams as the contact direction (similar to what is done for tube-to-tube contact elements, which are discussed in *Tube-to-Tube Contact Elements*). Four of the examples in *Figure 19* rely on the formulation with the cross product normal to resist penetrations, and the example on the lower right of *Figure 19* relies on

the formulation with the radial normal. The edge-to-edge contact formulation with the radial normal is applicable only to cases with some thickness contributing to the contact calculations.

The example shown in *Figure 20* involves compression of a spring modeled with beam elements. This example relies on the edge-to-edge contact formulation with a radial normal direction to resolve contact between adjacent spring coils, and it relies on the edge-to-surface contact formulation to resolve contact between the spring and other surfaces.

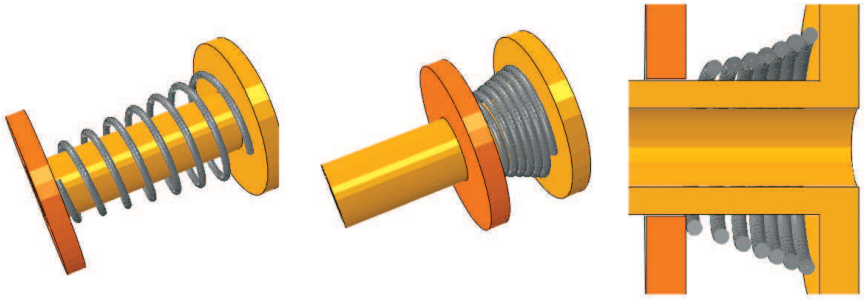


Figure 20: Compressing a spring modeled with beam elements.

The edge-to-edge contact formulation with a radial normal can involve the "exterior" of beam, shell, and solid feature edges and the "interior" of hollow beams, as shown in the example in *Figure 21*. This example involves a wire modeled with beam elements being wound onto a cylinder modeled with solid elements. The wire passes through a hollow cylindrical guide before coming onto the cylinder. The "radial" edge-to-edge formulation resolves contact between adjacent coils of the wire and also resolves contact between the wire and the interior of the hollow beam representing the guide. The edge-to-surface contact formulation resolves contact between the wire and the cylinder.

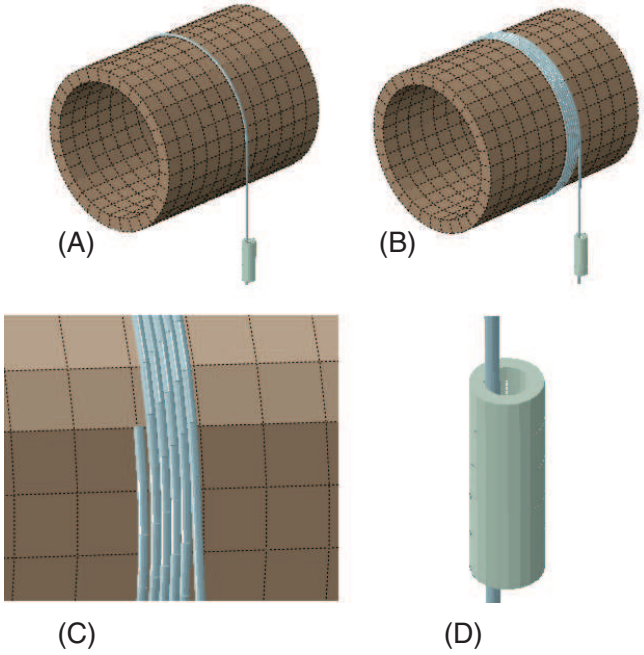


Figure 21: Winding a wire onto a cylinder.

The edge-to-edge contact formulation with a contact normal direction based on the cross product of the edge directions is applicable only while the edges are not nearly parallel. The edge-to-edge contact formulation with a radial contact normal direction is typically most applicable while contact edges are nearly parallel, but *Figure 22* shows an exception. The hollow beam is simultaneously in contact with the two other beams. The cross

product version of the edge-to-edge contact formulation resolves contact between the exterior of the hollow beam and the beam that is near the top of *Figure 22*. The radial version of the edge-to-edge contact formulation resolves contact between the interior of the hollow beam and the spiral-shaped beam, with the contact direction corresponding to the interior radial direction of the hollow beam. The radial version of the edge-to-edge contact formulation is effective in this case because individual segments of the spiral-shaped beam span relatively small arcs of the hollow tube.

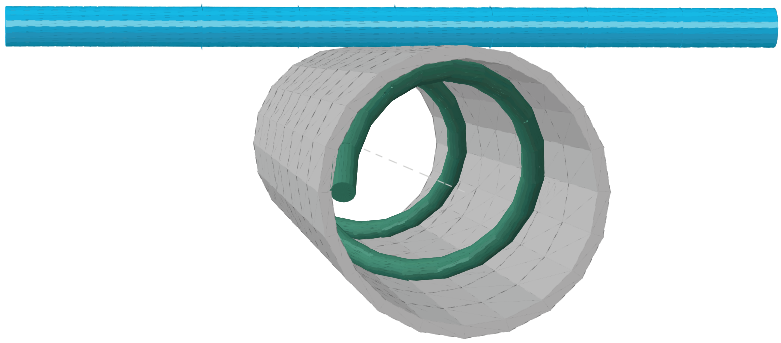


Figure 22: Edge-to-edge contact example with an internal beam spiral.

In addition to choosing to activate one or both types of edge-to-edge contact formulations, you must specify a feature angle criterion to activate feature and perimeter edges to participate in edge-to-edge contact. See *Feature Edges* for a discussion of controls related to which feature edges are considered for edge-to-edge contact. If only beam edges are present, specifying the contact formulation alone is sufficient.

Edge-to-edge contact formulations do not consider twisting of the beams. Beam-to-beam contact cannot be used to model contact between beam-like elements that share nodes with underlying solid or shell elements (for example, beam elements that are used to model stringers).

Input File Usage: Use the following option to activate both formulations for edge-to-edge contact:

**CONTACT FORMULATION*, TYPE=EDGE TO EDGE, FORMULATION=BOTH

Use the following option to deactivate edge-to-edge contact:

**CONTACT FORMULATION*, TYPE=EDGE TO EDGE, FORMULATION=NO (default)

Use the following option to activate the radial edge-to-edge contact formulation:

**CONTACT FORMULATION*, TYPE=EDGE TO EDGE, FORMULATION=RADIAL

Use the following option to activate the formulation based on the cross product of the edge directions for edge-to-edge contact:

**CONTACT FORMULATION*, TYPE=EDGE TO EDGE, FORMULATION=CROSS

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Contact Formulation**: toggle on **Edge-to-edge formulation assignment**, **Formulation**: select **Both**, **No**, **Radial**, or **Cross**

Vertex-to-Surface Contact Scenarios

The general contact algorithm can consider three-dimensional vertex-to-surface contact except on crack surfaces for enriched elements. *Figure 23* shows examples in which the vertex-to-surface contact formulation is most effective for resolving contact. The vertex-to-surface contact formulation is intended to avoid localized penetration of a node at a convex corner of a solid or shell/membrane surface or at an end point or kink of a beam/truss into a relatively smooth portion of another surface (which might be an analytical rigid surface). Most vertex nodes are along feature edges, although, for example, a node at the tip of a cone might satisfy the vertex node criteria. See *Vertex Nodes* for a discussion of the vertex node criteria. Vertex nodes are effectively treated as spherical in the vertex-to-surface formulation. The spherical radius of the contact vertex corresponds to the surface thickness at the node.

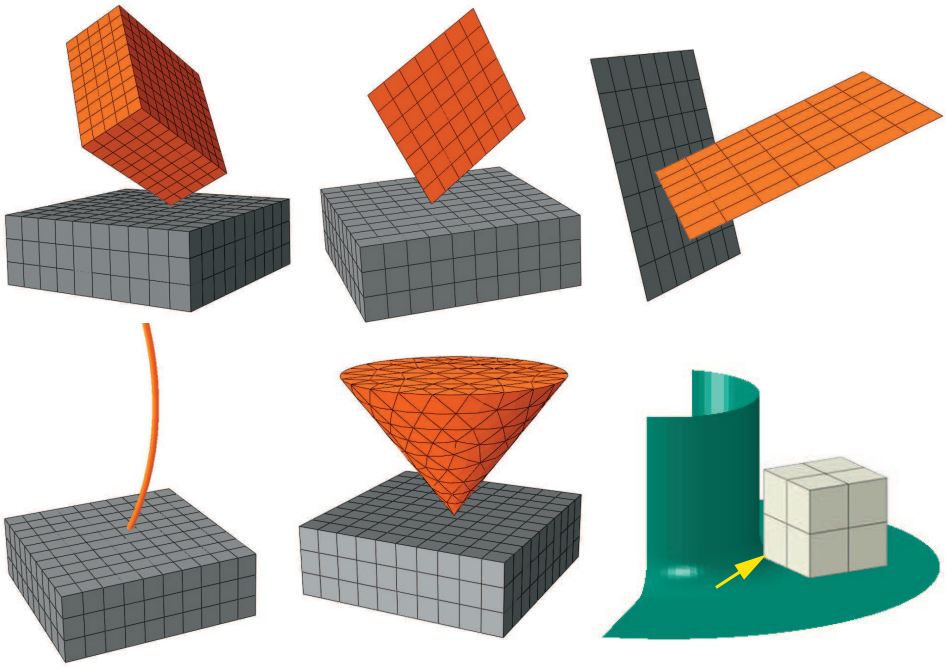


Figure 23: Vertex-to-surface contact examples.

The bottom-right example in *Figure 23* shows a vertex node of an element-based surface in contact with an analytical rigid surface. The corners of an analytical rigid surface do not act as vertices in the vertex-to-surface formulation. Convergence behavior can be enhanced by avoiding feature edges and corners in analytical rigid surfaces used for contact and instead using smooth analytical rigid surfaces with continuously varying surface normal directions for contact.

Output

Output variables associated with contact fall into two categories: nodal variables (sometimes called constraint variables) and whole surface variables. In addition, Abaqus outputs an array of diagnostic information associated with contact interactions, as discussed in *Contact Diagnostics in an Abaqus/Standard Analysis*, and internal surfaces generated for general contact.

For more detailed discussions of variables associated with thermal, electrical, and pore fluid analyses, see the sections on the related contact properties in *Contact Property Models*.

General Contact Domain and Component Surfaces in Abaqus/Standard

Abaqus/Standard generates the following internal surfaces associated with general contact:

- `General_Contact_Faces`,
- `General_Contact_Edges`,
- `General_Contact_Vertices`,
- `General_Contact_Faces_k`,
- `General_Contact_Edges_k`, and
- `General_Contact_Vertices_k`,

where `k` corresponds to an automatically assigned "component number." The three internal surfaces for general contact without a component number contain all surface faces, all feature edges, and all vertices, respectively, included in the general contact domain.

Each feature edge component surface, `General_Contact_Edges_k`, has a subset of face edges (satisfying the feature edge criteria) of the corresponding face component surface, `General_Contact_Faces_k`. Each vertex component surface, `General_Contact_Vertices_k`, has a subset of vertices (satisfying the vertex criteria) of the corresponding face component surface, `General_Contact_Faces_k`. The face component surfaces have no nodes in common with each other, except if beams and trusses are part of the contact domain that might share nodes with other faceted component surfaces. By default, a lowered-numbered component surface will act as a main surface to a higher-numbered component surface for the surface-to-surface and the radial version of the edge-to-edge formulations. Component numbers do not influence what is considered by the edge-to-surface, vertex-to-surface, and cross version of the edge-to-edge formulations. A component surface consisting of beam and truss elements will act as a main surface in the edge-to-surface formulation if half of the average element radius is larger than the average smallest facet length of the faceted component surface. Component surfaces are referred to in diagnostic messages for all formulation types.

Abaqus/Standard also generates internal surfaces associated with general contact when material names are used to identify regions where nondefault contact properties or surface properties are assigned, as discussed in *Assigning Contact Properties* and *Assigning Surface Properties*. These internal surfaces are named `_MATSURF_Material Name_`, where *Material Name* corresponds to the name of the material specified for the property assignment.

Internal surfaces can be viewed using display groups in the Visualization module of Abaqus/CAE. Internal surface names generated by Abaqus/Standard should not be used in model definitions.

Nodal Contact Variables

Nodal contact variables can be contoured on contact surfaces in the Visualization module of Abaqus/CAE. Nodal contact variables include contact pressure and force, frictional shear stress and force, relative tangential motion (slip) of the surfaces during contact, clearance between surfaces, heat or fluid flux per unit area, and fluid pressure. Many of the nodal contact variables written to the output database (`.odb`) file are often available for all contact nodes, regardless of whether they act as secondary or main nodes. Other nodal contact variables are available only at nodes acting as secondary nodes. Most contact output to the data (`.dat`) file, results (`.fil`) file, and the utility subroutine `GETVRMAVGATNODE` is associated with individual constraints. For contact output to the output database (`.odb`) file, some filtering is applied to reduce contact output noise.

Contact Pressure

The contact pressure distribution is of key interest in many Abaqus analyses. You can view the contact pressure on all contact surfaces except for analytical rigid surfaces and discrete rigid surfaces based on rigid-type elements (the latter restriction does not apply to general contact). You can view a contour plot of the contact pressure error indicator next to a contour plot of the contact pressure to gain perspective on local accuracy of the contact pressure

solution in regions where the contact pressure solution is of interest (see *Selection of Error Indicators Influencing Adaptive Remeshing*, for further discussion of error indicator output).

In some cases you might observe the contact pressure extending beyond the actual contact zone due to the following factors:

- The contour plots are constructed by interpolating nodal values, which can cause nonzero values to appear within portions of facets outside of the contact region. For example, this effect is often noticeable at corners, such as when two same-sized, aligned blocks are in contact—if the contact surfaces wrap around the corners, the contact pressure contours will extend slightly around the corners.
- Abaqus/Standard outputs postprocessed contact stresses to the output database. During postprocessing, nodal contact stresses are calculated as weighted averages of values associated with active contact constraints in which the node participates. For example, a main node can participate in multiple constraints whose connectivities contain the node. Similarly, a secondary node can participate in multiple constraints from different formulations (such as surface-to-surface, edge-to-surface, and vertex-to-surface) at secondary node locations that are corners and edge features.

  The weighting depends on the contact constraint area and a scaling factor based on the strength of the participation of the node in a constraint. The weighted averaging is intended to reduce contact stress noise. Modifications are made for calculating weightings, for example, at the corner nodes of quadratic faces (with zero consistent nodal areas) and main nodes that are outside the active contact region but participate in contact weakly through an active contact constraint. These modifications also have a filtering effect in terms of reducing contact stress values reported for nodes on the fringe of the active contact region. For such locations, contact nodal areas that are simply cumulative scaled constraint areas across constraints in which a node participates do not have much bearing on contact stress values.

  In addition to averaging and filtering, contact stresses are also smoothed during the postprocessing operations. However, this filtering and subsequent smoothing are not "perfect" and can result in the contact zone size appearing somewhat exaggerated. Similarly, contact status output is also affected at nodes that lie on the fringe of the active contact region. In such cases, the contact status might be reported as closed at nodes in the exaggerated region even though it is open.

Due to these factors, trying to infer the contact force distribution from the contact stress distribution can be somewhat misleading. Instead, you can request nodal contact force output, which accurately represents the contact force distribution present in the analysis.

Contact Stresses due to Edge-to-Surface, Edge-to-Edge, and Vertex-to-Surface Interactions

For edge-to-surface contact and for edge-to-edge contact with the radial formulation where the active contact is along a line, the output variable CLINELOAD can be requested to the output database (`.odb`) in Abaqus/Standard. This contact load has units of force per length and is mesh independent. Contact stresses (in units of force per area) solely due to edge-to-surface contact (CSTRESSETOS) can be output for visualizing regions where the edge-to-surface constraints are active. The edge-to-surface formulation computes contact stresses in units of force per area by dividing contact force per edge length by a representative surface facet length. Since the contact area depends on the mesh size, edge-to-surface contact stresses are mesh dependent. For edge-to-edge contact using the cross product formulation where the active contact region is idealized as a point, the mesh-independent output variable CPOINTLOAD (with units of force) can be requested.

For vertex-to-surface contact, the mesh-independent output variable CPOINTLOAD (with units of force) can be requested to the output database (`.odb`) in Abaqus/Standard.

Contact stresses (CSTRESS) contain contributions from surface-to-surface, edge-to-surface, edge-to-edge, and vertex-to-surface constraints, if active. While accumulating contributions from edge-to-surface, edge-to-edge, and vertex-to-surface contact constraints, the constraint values are divided by either a representative surface facet length or its squared value to appropriately scale them to have units of force per area.

Edges and vertices represent a discontinuity in the surface smoothness, and the true contact stress solution near an edge or a vertex is commonly characterized by a strong gradient. Subsequently, error indicator output for contact stresses (CSTRESSERI) are typically quite high and acceptable for regions in which constraints involving edges and vertices are significant.

Whole Surface Variables

Whole surface variables (see *Mechanical Analysis–Whole Surface Quantities*) are only marginally supported for general contact in Abaqus/Standard because these variable are associated with the overall general contact domain by default rather than individual surfaces associated with general contact. The only way to limit whole surface variables to be affected by a portion of the general contact domain is to specify a node set in the output request. Whole surface variables are computed as sums over all nodes (or optionally limited to a particular node set) of general contact while acting as secondary nodes. For example, CFN is the total force acting on secondary nodes due to contact pressure. CFN and other whole surface variables for general contact are typically of little utility, because contributions to the variable from different interactions within general contact will often cancel one another and the net result will typically depend on internal assignments of main and secondary roles.

Requesting Output

Certain contact variables must be requested as a group. For example, to output the clearance between surfaces (COPEN), you must request the variable CDISP (contact displacements). CDISP outputs both COPEN and CSLIP (tangential motion of the surfaces during contact). A complete listing of available contact variables and identifiers is given in *Abaqus/Standard Output Variable Identifiers*.

Output requests can be limited by specifying a node set containing a subset of the nodes acting as secondary nodes for some general contact interactions. Instructions on forming these output requests are available in the following sections:

- To request output to the data (`.dat`) file, see *Surface Output from Abaqus/Standard*.
- To request output to the output database (`.odb`) file, see *Writing Surface Output to the Output Database*.

Output of Tangential Results

Abaqus reports the values of tangential variables (frictional shear stress, viscous shear stress, and relative tangential motion) with respect to the local tangent directions defined on the surfaces. The local tangent directions CTANDIR1 and CTANDIR2 can be output by requesting the generic output variable CTANDIR. The definition of local tangent directions is explained in *Local Tangent Directions on a Surface*. These directions do not always correspond to the global coordinate system, and they rotate with the contact pair in a geometrically nonlinear analysis.

Abaqus/Standard calculates tangential results at each constraint point by taking the scalar product of the variable's vector and a local tangent direction, $\mathbf{t}_1$ or $\mathbf{t}_2$, associated with the constraint point. The number at the end of a variable's name indicates whether the variable corresponds to the first or second local tangent direction. For example, CSHEAR1 is the frictional shear stress component in the first local tangent direction, while CSHEAR2 is the frictional shear stress component in the second local tangent direction.

Definition of Accumulated Incremental Relative Motion (Slip)

Abaqus/Standard defines the incremental relative motion (also known as slip) as the scalar product of the incremental relative nodal displacement vector and a local tangent direction. The incremental relative nodal displacement vector measures the motion of a secondary node relative to the motion of the main surface. The incremental slip is accumulated only when the secondary node is contacting the main surface. The sums of all such incremental slips during the analysis are reported as CSLIP1 and CSLIP2. Details about the calculation of this quantity can be found in *Small-sliding interaction between bodies*, *Finite-sliding interaction between deformable bodies*, and *Finite-sliding interaction between a deformable and a rigid body*.

Output for Beams with Actual Beam Cross-Section Representation and Shells with Actual Shell Perimeter Representation

When you enable actual beam cross-section representation for beams (see *Contact Surface Representation for Beams*) or actual shell perimeter representation for shells (see *Contact Surface Representation for Shells*), Abaqus uses an automatically generated mesh of surface elements to represent the contact surface. Output variables associated with contact are available on the surface elements for postprocessing. They are not available on the corresponding beam or shell elements. Output variables such as stress and strain at integration points are available on the beam or shell elements rather than the surface elements. For an example, see *Visualizing beam outputs with actual beam cross-section representation for contact*.

Extending the Range for Which Contact Opening Output Is Provided for Gaps

To reduce computational costs, detailed computations to monitor potential points of interaction are avoided by default where surfaces are separated by a distance greater than the minimum gap distance at which contact forces (or thermal fluxes, etc.) might be transmitted. Therefore, contact opening (COPEN) output is typically not provided where surfaces are opened by more than a small amount compared to surface facet dimensions. You can extend the range for which Abaqus/Standard provides contact opening output; COPEN will be provided up to gap distances equal to a specified "tracking thickness." Using this control might increase computational cost due to extra contact tracking computations, especially if you specify a large tracking thickness value.

Input File Usage: **SURFACE INTERACTION*, TRACKING THICKNESS=*value*

Abaqus/CAE Usage: You cannot adjust the default tracking thickness in Abaqus/CAE.

Whole Model Contact-Related Energy Variables

The contact-related energy variables, shown in *Table 4*, are available in Abaqus/Standard (see *Abaqus/Standard Output Variable Identifiers*). An example of using the contact-related energies is provided in *Energy computations in a contact analysis*.

Table 4: Contact-related energy output variables.

Description		Output variable
Frictional dissipation		ALLFD
Elastic contact energy	Energy stored among all penalty springs and "softened" contact constraints associated with normal contact constraints	ALLCCEN
	Energy stored among all penalty springs associated with tangential contact constraints	ALLCCET
	Energy stored among all penalty springs and "softened" contact constraints associated with normal and tangential contact constraints (equal to the sum of ALLCCEN and ALLCCET)	ALLCCE
Energy dissipation associated with contact stabilization and contact damping	Normal contact direction for the whole model	ALLCCSDN
	Tangential contact direction for the whole model	ALLCCSDT
	Whole model (equal to the sum of ALLCCSDN and ALLCCSDT)	ALLCCSD
Energy associated with contact constraint "discontinuity work"	Accounts for the portion of the work done by contact forces when contact conditions change that is not accounted for by other contact energy variables	ALLCCDW

The output variables ALLSD and ALLVD also account for dissipative energies associated with contact stabilization and contact damping.

The elastic contact energies and dissipative energies associated with contact stabilization and contact damping are associated with numerical effects that would be zero in idealized situations, such as infinite penalty stiffness or zero stabilization stiffness. Significant values of these output variables compared to other physically based energies in a model, such as internal energy (ALLIE), are sometimes indicative of solution inaccuracy. The contact constraint discontinuity work will tend to zero as the time increment size becomes very small. However, as discussed in *Energy computations in a contact analysis*, it is quite common for ALLCCDW to have a significant value without causing solution inaccuracy.

The modified external work (ALLWK + ALLCCDW) is often representative of the physical external work in contact problems in terms of being equal to the sum of the stored and dissipated energies (see *Energy computations in a contact analysis*). Consider a particular contact constraint having a gap distance, g_i, in one increment and becoming closed with contact force, f_{i+1}, in the next increment (see *Figure 24*). A trapezoidal rule for integrating the work done by the contact force multiplies the average force by the relative incremental motion. In this case, the resulting contribution to ALLCCDW is negative $1/2 * g_i * f_{i+1}$. This energy contribution is nonphysical and would disappear in the numerics as the time increment tends to zero. When contact opens up, similar behavior happens with sign reversals. Numerical integration for ALLWK is also limited with respect to accounting accurately for sudden changes in external forces. Summing ALLWK and ALLCCDW often cancels the respective nonphysical energy contributions, and the net effect on the total energy balance ETOTAL is zero.

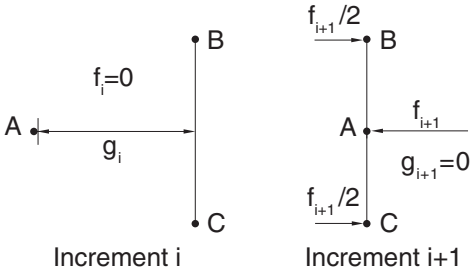


Figure 24: One contact point example to illustrate contributions to ALLCCDW.

Surface Properties for General Contact in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About General Contact in Abaqus/Standard*
- **CONTACT*
- **SURFACE PROPERTY ASSIGNMENT*
- *Specifying surface property assignments for general contact*

Overview

Surface property assignments:

- can be used to specify geometric corrections for regions of a surface;
- can be used to change the contact thickness used for regions of a surface based on structural elements or to add a contact thickness for regions of a surface based on solid elements;
- can be used to specify surface offsets for regions of a surface based on shell, membrane, rigid, and surface elements; and
- can be applied selectively to particular regions within a general contact domain.

Assigning Surface Properties

You can assign nondefault surface properties to surfaces involved in general contact interactions. These properties are considered only when the surfaces are involved in general contact interactions; they are not considered when the surfaces are involved in other interactions such as contact pairs. The general contact algorithm does not consider surface properties specified as part of the surface definition. The regions with nondefault surface properties are identified with surface names or material names. For example, surface property `SurfProp_A` can assign a nondefault surface thickness to surface `Surf_1` or to the surface whose underlying elements have a section assignment with material `Rubber`. Material names cannot be used to control beam smoothing or to assign geometric corrections.

Surface properties for general contact in Abaqus/Standard are assigned at the beginning of an analysis and cannot be modified across steps.

The surface names used to specify the regions with nondefault surface properties do not have to correspond to the surface names used to specify the general contact domain. In many cases the contact interaction will be defined for a large domain, while nondefault surface properties will be assigned to a subset of this domain. Any surface property assignments for regions that fall outside the general contact domain will be ignored. The last assignment will take precedence if the specified regions overlap.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY

This option must be used in conjunction with the **CONTACT* option and should appear at most once for each value of the PROPERTY parameter discussed below; the data line can be repeated as often as necessary to assign surface properties to different regions.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**

Geometry-Based Corrections for Curved Surfaces

Contact calculations are based on unsmoothed, faceted representations of the finite element surfaces in a general contact domain. For curved surfaces, the finite element representation of a surface can deviate significantly from the original geometry that was used to generate the finite element mesh. A contact smoothing technique based on the original geometric representation can be employed for a more realistic simulation of contact interactions between curved surfaces, resulting in improved accuracy of stresses and distances between the contacting surfaces. This contact smoothing technique is discussed in more detail in *Smoothing Contact Surfaces in Abaqus/Standard*.

Surface Thickness

The default surface thickness is equal to the original parent element thickness. Alternatively, you can specify a value for the surface thickness or a thickness scaling factor. A nonzero thickness can be assigned to solid element surfaces; for example, to model the effect of a finite thickness surface coating.

Using the Original Parent Element Thickness

The default surface thickness is equal to the original parent element thickness.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=THICKNESS
surface or *material*, ORIGINAL (default), , SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The fourth entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Surface thickness assignments: Edit**:
Select surface or material, click the arrows to transfer surface or material to list of thickness
assignments, and enter ORIGINAL in the **Thickness** column.

Specifying a Value for the Surface Thickness

You can specify the surface thickness value directly.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=THICKNESS
surface or *material*, *value*, , SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The fourth entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Surface thickness assignments: Edit**:

Select surface or material, click the arrows to transfer surface or material to list of thickness
assignments, and enter a value for the surface thickness magnitude in the **Thickness** column.

Applying a Scale Factor to the Surface Thickness

You can apply a scale factor to any value of the surface thickness. For example, if you specify that the original parent element thickness should be used for `surf1` and apply a scale factor of 0.5, a value of one half the original parent element thickness will be used for `surf1` when it is involved in a general contact interaction (all other surfaces included in the general contact domain will use the default original parent element thickness). Scaling the surface thickness in this way can be used to avoid initial overclosures in some situations. Abaqus/Standard will automatically adjust surface positions to resolve initial overclosures (see *Contact Initialization for General Contact in Abaqus/Standard*) associated with general contact. However, if nodal position adjustments are undesirable (for example, if they would introduce an imperfection in an otherwise flat part, resulting in an unrealistic buckling mode), you may prefer to reduce the surface thickness and avoid the overclosures entirely.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=THICKNESS
surface or *material*, *value or label*, *scale_factor*, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The fourth entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**:
Surface Properties: **Surface thickness assignments: Edit**:
Select surface or material, click the arrows to transfer surface or material to list of thickness
assignments, and enter a **Scale Factor**.

Surface Offset

A surface offset is the distance between the midplane of a thin body and its reference plane (defined by the nodal coordinates and element connectivities). It is computed by multiplying the offset fraction (specified as a fraction of the surface thickness) by the surface thickness and the element facet normal. This defines the position of the midsurface and, thus, the position of the body with respect to the reference surface; the coordinates of the nodes on the reference surface are not modified. Surface offsets can be specified only for surfaces defined on shell and similar elements (i.e., membrane, rigid, and surface elements). Surface offsets specified for other elements (e.g., solid or beam elements) will be ignored. By default, surface offsets specified in element section definitions will be used in the general contact algorithm.

You specify the surface offset as a fraction of the surface thickness. The surface offset fraction can be set equal to the offset fraction used for the surface's parent elements or to a specified value. Surface offsets specified for general contact do not change the element integration.

Input File Usage: Use the following option to use the surface offset fraction from the surface's parent elements (default):

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=OFFSET FRACTION
surface or *material*, ORIGINAL, SURFACE (default) or MATERIAL

Use the following option to specify a value for the surface offset fraction:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=OFFSET FRACTION
surface or *material*, *offset*, SURFACE (default) or MATERIAL

The offset can be specified as a value or a label (SPOS or SNEG). Specifying SPOS is equivalent to specifying a value of 0.5; specifying SNEG is equivalent to specifying a value of −0.5.

The third entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**:
Surface Properties: **Shell/Membrane offset assignments: Edit**:
Select surface or material, and click the arrows to transfer surface or material to list of offset
assignments.
In the **Offset Fraction** column, enter ORIGINAL to use the surface offset fraction from the surface's parent elements, enter SPOS to use a surface offset fraction of 0.5, enter SNEG to use a surface offset fraction of −0.5, or enter a value for the surface offset fraction.

Feature Edges

Feature edges of a model are defined on beam and truss elements and on edges of faces (perimeter and otherwise) of solid and structural elements. General contact in Abaqus/Standard includes an edge-to-surface contact formulation and an edge-to-edge contact formulation (as supplements to the surface-to-surface formulation), as discussed in *About General Contact in Abaqus/Standard*. By default, the edge-to-surface contact formulation considers “edges” of beam and truss elements, perimeter edges, and edges corresponding to initial geometric feature angles of 45° and higher. You can control the feature edge criterion globally or locally for both edge-to-surface and edge-to-edge contact. Feature edge criteria have no effect on “edges” of beam and truss elements—they are activated by their inclusion in the contact domain.

Some aspects of the contact property assignment options apply only to the surface-to-surface formulation (see *Contact Properties for General Contact in Abaqus/Standard* for further discussion of contact properties for general contact). The edge-to-surface and edge-to-edge formulations always use the penalty enforcement method and only involve displacement degrees of freedom. For example, the edge-to-surface formulation or the cross edge-to-edge formulation does not contribute to thermal gap conductance across a contact interface.

Specifying a Cutoff Feature Angle

The feature angle is the angle formed between normals of two facets connected to an edge. The angles between facets are based on the initial configuration. A negative angle results at concave meetings of facets; therefore, these edges are never included in the contact domain. *Figure 1* shows some examples of how the feature angle is calculated for different edges.

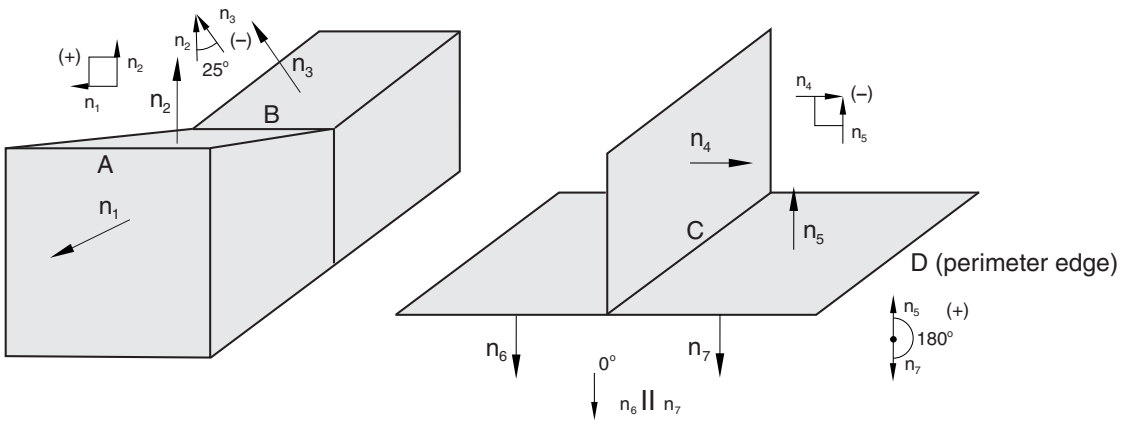


Figure 1: Calculating the feature angle.

The feature angle for edge A is 90° (the angle between $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$); the feature angle for edge B is −25° (the angle between $\boldsymbol{n_2}$ and $\boldsymbol{n_3}$). Edge C forms a T-intersection with three facets (shown in two dimensions in *Figure 2*); its feature angles are 0°, −90°, and −90°.

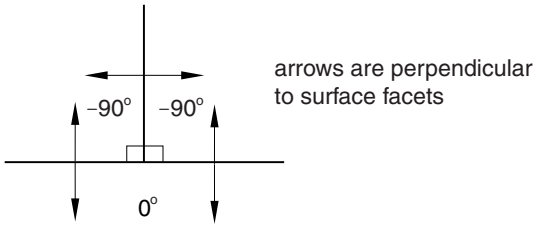
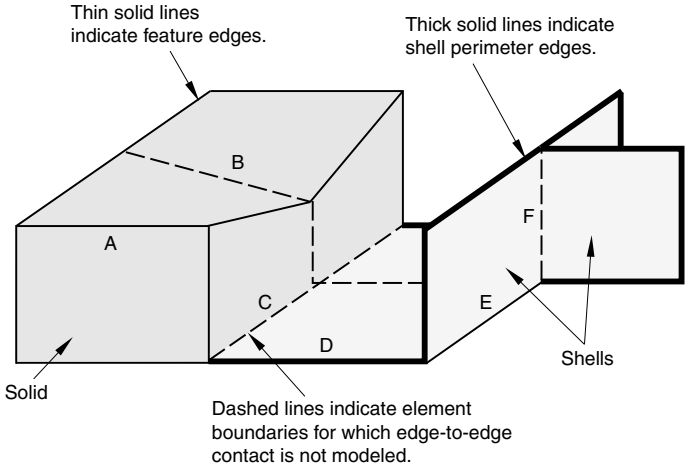


Figure 2: Feature angles for a T-intersection.

Perimeter edges (for example, edge D in *Figure 1*) can be thought of as a special type of feature edge where the feature angle is 180°.

If a feature angle criterion is in effect (by default or because you specified it), geometric edges of solid and shell bodies with feature angles greater than or equal to the specified angle are included in the general contact domain. The contact inclusion and exclusion options (discussed in *About General Contact in Abaqus/Standard*) apply to the surface-to-surface contact formulation, the edge-to-surface contact formulation, and the edge-to-edge contact formulation (and further control which portions of surfaces may interact with either formulation). The sign of the feature angle is considered when determining whether or not a geometric feature edge should be included in the general contact domain. For example, if a cutoff feature angle of 20° were specified, edge A would be activated as a feature edge in the contact model (because the feature angle of 90° is greater than the cutoff of 20°) but edges B and C would not be activated (because the feature angle at edge B is −25° and the maximum feature angle at edge C is 0°, which are both less than the cutoff of 20°). The cutoff feature angle cannot be set to less than 0° or more than 180°. Specifying a small cutoff feature angle (for example, less than 20°) may considerably increase run time without a major impact on the results compared to a larger cutoff angle (> 20°). The default feature angle cutoff for edge-to-surface contact is 45°, while the default is not to include feature edges in edge-to-edge contact. The criterion for including edges for edge-to-surface contact can be different from the criterion for including edges for edge-to-edge contact; the two are completely independent.

Figure 3 illustrates further how the feature angle is used to determine which geometric feature edges are activated in the general contact domain.



Edge	Largest feature angle at edge	Other feature angles at edge
A	approximately $+105^{o}$	none
B	approximately -30^{o}	none
C	0^{o}	-90^{o}
D	$+180^{o}$	none
E	$+90^{o}$	-90^{o}
F	0^{o}	$-90^{o}, -90^{o}$

Figure 3: Feature edges activated in the general contact domain for a cutoff feature angle of 45°.

The table to the right of the figure lists the feature angle values for various edges in the model. Edges connected to shell facets, but not on the shell perimeter, have more than one corresponding feature angle. The largest feature angle at an edge is compared to the default or specified cutoff feature angle. For example, if the default cutoff feature angle of 45° is in effect, edges A, D, and E would be considered for edge-to-surface contact, while edges B, C, and F would be ignored for edge-to-surface contact.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA
surface or *material*, *feature_angle_value_edge_to_surface*, ,
feature_angle_value_edge_to_edge, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The fifth entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Use the following options to specify the cutoff feature angle for edge-to-surface contact:

Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select the surface or material, click the arrows to transfer the surface or material to the list of feature assignments, and enter a numerical value for the cutoff feature angle (in degrees) in the **Primary Feature Edge Criteria** column, and the cutoff feature angle for edge-to-edge contact in the **Secondary Feature Edge Criteria** column.

Specifying That Only Perimeter Edges Should Be Activated

You can specify that only perimeter edges should be considered by the edge-to-surface and/or edge-to-edge formulation globally or in a local region. Perimeter edges occur on “physical” perimeters of shell elements and on “artificial” edges that occur when a subset of exposed facets on a body are included in the general contact domain. The classification of an edge as being on the perimeter of the contact domain (or as a geometric edge with a particular feature angle) is based on the contact inclusion and contact exclusion definitions and the mesh

characteristics. When structural elements share nodes with continuum elements, the perimeter edges will not be activated on the structural elements because the criterion to designate them as such is no longer satisfied.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA
surface or *material*, PERIMETER EDGES, , PERIMETER EDGES, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The fifth entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Use the following options to specify that only perimeter edges should be included for edge-to-surface contact:

Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select the surface or material, click the arrows to transfer the surface or material to the list of feature assignments, and enter PERIMETER in the **Primary Feature Edge Criteria** column, and in the **Secondary Feature Edge Criteria** column to specifying that only perimeter edges should be included for edge-to-edge contact

Specifying That Feature Edges Should Not Be Included

You can specify that no edges should be considered by the edge-to-surface formulation globally or in a local region. However, doing so does not deactivate “contact edges” associated with beam and truss elements. By default, feature edges are not included for edge-to-edge contact.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA
surface or *material*, NO FEATURE EDGES, , , SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The fifth entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Use the following options to specify that no feature edges are included for edge-to-surface contact:

Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select the surface or material, click the arrows to transfer the surface or material to the list of feature assignments, and enter NONE in the **Primary Feature Edge Criteria** column

Vertex Nodes

Vertex nodes of a model are defined on convex corners of shell and structural surfaces and end points and kinks in beam and truss surfaces. Vertex nodes are eligible to participate as vertices in the vertex-to-surface contact

formulation for general contact in Abaqus/Standard. The vertex-to-surface contact formulation is discussed in *About General Contact in Abaqus/Standard*.

Some aspects of the contact property assignment options apply only to the surface-to-surface formulation (see *Contact Properties for General Contact in Abaqus/Standard* for further discussion of contact properties for general contact). The edge-to-surface and vertex-to-surface formulations always use the penalty enforcement method and involve only displacement degrees of freedom. For example, the vertex-to-surface formulation does not contribute to thermal gap conductance across a contact interface.

Usually, only a small subset of surface nodes satisfy one of the criteria for being a vertex node. The following algorithm is used to determine if a node satisfies the vertex criterion for a convex corner of a solid or shell-like surface, based on the original configuration:

- Abaqus/Standard computes an average surface normal direction for the node and an associated plane that passes through the node and is perpendicular to this direction. Two examples of such normal directions and planes are shown in *Figure 4*.
- The node is considered a vertex node if each surface edge emanating from this node is "inside" of this plane (that is, on the opposite side of this plane as the average surface normal direction vector) and forms an angle greater than or equal to the vertex angle threshold with this plane, by default. For example, the example node on the right of *Figure 4* has all adjacent surface edges on the "inside" of the plane perpendicular to the average surface normal direction. For the other example node in *Figure 4* for which an average normal direction and corresponding plane is shown, two adjacent surface edges lie outside of this plane (on the same side as the outward normal direction vector), so this node would not satisfy this vertex node criterion.

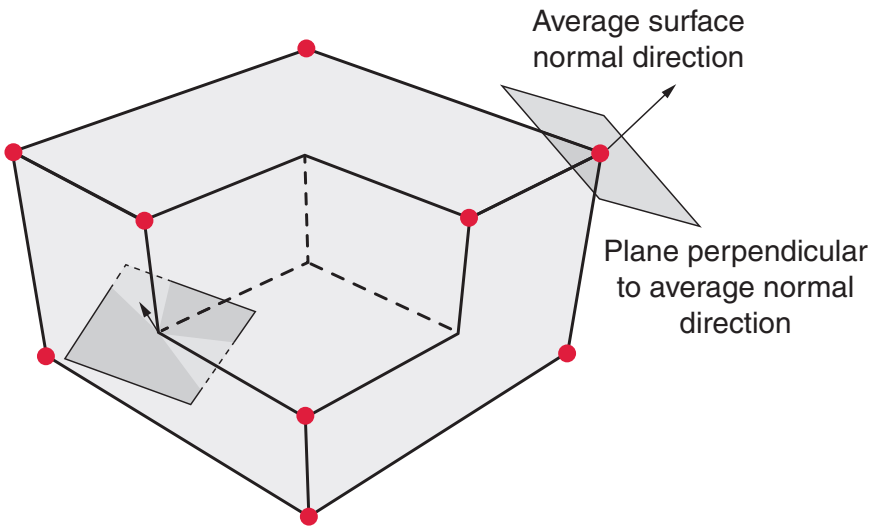


Figure 4: Examples of average surface normal directions and corresponding planes associated with vertex node criterion.

The following additional vertex criterion is applied at nodes of perimeter feature edges, based on the original configuration:

- Abaqus/Standard computes an average outward perimeter direction for the node and an associated plane that passes through the node and is perpendicular to this direction.
- The node is considered a vertex node if each feature edge connected to the node is inside of this plane and forms an angle greater than or equal to the vertex angle threshold with this plane, by default.

The following vertex criterion is applied at nodes of feature edges associated with beams and trusses, based on the original configuration:

- Abaqus/Standard computes an average edge direction for the node and an associated plane that passes through the node and is perpendicular to this direction.
- If the average edge direction is zero, the node is not a vertex node; otherwise, the node is considered a vertex node if each feature edge connected to the node forms an angle greater than or equal to the vertex angle threshold with this plane, by default.

The default vertex angle threshold is 20°. The circular dots in *Figure 5* represent examples of nodal locations that would satisfy one of the vertex node criteria, with the default vertex angle threshold in effect.

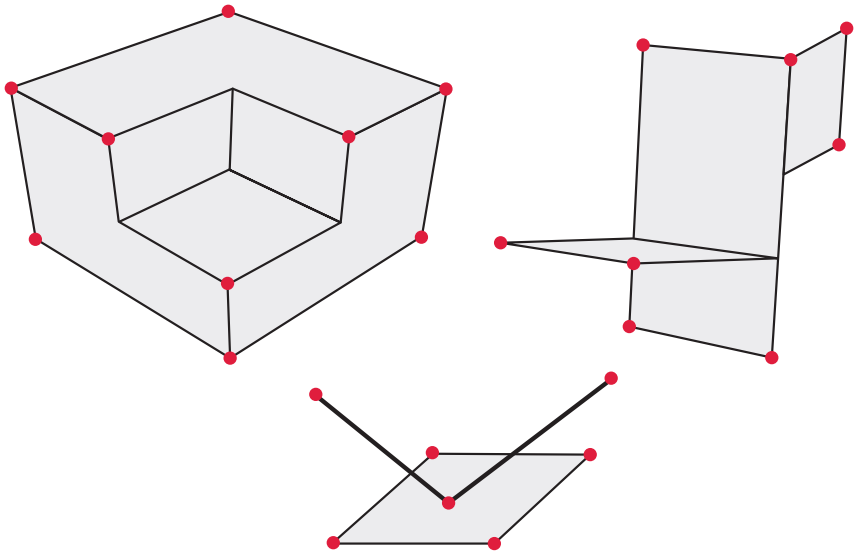


Figure 5: Examples of vertex locations represented by circular dots.

Specifying That All Vertex Nodes Should Be Included

You can specify that all vertex nodes should be considered by the vertex-to-surface formulation globally or in a local region.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=VERTEX CRITERIA
surface or *material*, ALL VERTICES, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The third entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Use the following options to specify that all vertex nodes should be included for vertex-to-surface contact:

Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Surface vertex criteria assignments: Edit**:
Select the surface or material, click the arrows to transfer the surface or material to the list of vertex criteria assignments, and enter ALL_VERTICES in the **Vertex Criteria** column.

Specifying a Vertex Angle Threshold

You can control the vertex criteria globally or locally.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=VERTEX CRITERIA
surface or *material*, *vertex_angle_threshold*, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The third entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Surface vertex criteria assignments: Edit**:
Select the surface or material, click the arrows to transfer the surface or material to the list of vertex criteria assignments, and enter a numeric value for vertex anglr threshold (in degrees) in the **Vertex Criteria** column.

Specifying That Vertex Nodes Should Not Be Included

You can specify that no vertex nodes should be considered by the vertex-to-surface formulation globally or in a local region. However, doing so does not deactivate “contact edges” associated with beam and truss elements.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=VERTEX CRITERIA
surface or *material*, NO VERTICES, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The third entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Use the following options to specify that no vertex nodes should be included for vertex-to-surface contact:

Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Surface vertex criteria assignments: Edit**:
Select the surface or material, click the arrows to transfer the surface or material to the list of vertex criteria assignments, and enter NO_VERTICES in the **Vertex Criteria** column.

Contact Properties for General Contact in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About General Contact in Abaqus/Standard*
- *About Mechanical Contact Properties*
- *Contact Pressure-Overclosure Relationships*
- *Contact Damping*
- *Frictional Behavior*
- **CONTACT*
- **CONTACT PROPERTY ASSIGNMENT*
- **SURFACE INTERACTION*
- *Specifying and modifying contact property assignments for general contact*

Overview

Contact properties:

- define the surface interaction models that govern the behavior of surfaces when they are in contact; and
- can be applied selectively to particular regions within a general contact domain.

Assigning Contact Properties

The default contact property model in Abaqus/Standard assumes “hard” contact in the normal direction, no friction, no thermal interactions, etc. You can assign a nondefault contact property definition (surface interaction) to specified regions of the general contact domain. These regions are identified with surface names or material names. For example, contact property `ContProp_A` can be assigned to the combination of surface `Surf_1` and the surface whose underlying elements have a section assignment with material `Rubber`. Surfaces associated with `Surf_1` and `Rubber` can span multiple parts.

Contact properties for general contact in Abaqus/Standard are assigned at the beginning of the analysis and cannot be modified across steps, with an exception for changes to the friction model, as discussed below.

The surface names used to specify the regions where nondefault contact properties should be assigned do not have to correspond to the surface names used to specify the general contact domain. In many cases the contact interaction will be defined for a large domain, while nondefault contact properties will be assigned to a subset of this domain. Any contact property assignments for regions that fall outside of the general contact domain will be ignored. The last assignment will take precedence if the specified regions overlap.

Input File Usage: **CONTACT PROPERTY ASSIGNMENT*
surface_1 or *material_1*, *surface_2* or *material_2*, *interaction_property_name*,
SURFACE (default) or MATERIAL, SURFACE (default) or MATERIAL

This option must be used in conjunction with the **CONTACT* option and should appear at most once; the data line can be repeated as often as necessary to assign contact properties to different regions.

The fourth and fifth entries indicate whether the first and second entries, respectively, refer to surface or material names. If omitted, Abaqus assumes that a surface name is used.

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed. If the second entry is omitted or is the same as the first entry, contact between the first named or implied surface and itself is assumed. Surfaces can span multiple unattached bodies, so self-contact of a surface is not limited to contact of a single body with itself. If the interaction property name is omitted, the unnamed set of default contact properties in Abaqus/Standard is assumed. If an interaction property name is specified, it must also appear as the value of the NAME parameter on a **SURFACE INTERACTION* option in the model portion of the input file.

Abaqus/CAE Usage: Use the following options to assign a global contact property to the entire general contact domain:

Interaction module: **Create Interaction**: **General contact (Standard)**: **Contact Properties**: **Global property assignment:** *interaction_property_name*

Use the following options to assign contact properties to individual surface pairs:

Interaction module: **Create Interaction**: **General contact (Standard)**: **Contact Properties**: **Individual property assignments: Edit**: select the combinations of surfaces and materials and the contact property in the columns on the left, and click the arrows in the middle to transfer them to the list of contact property assignments

In Abaqus/CAE you must assign a global contact property; Abaqus/CAE does not assume a default contact interaction property. Contact properties assigned to individual surface pairs override the global assignment.

Changing Friction Properties during an Analysis

The friction properties associated with a given named surface interaction definition can be modified in any particular step of an Abaqus/Standard analysis, as discussed in *Changing Friction Properties during an Abaqus/Standard Analysis*.

Example

The following contact property assignments are specified below as model data in a general contact analysis:

- a global assignment of `contProp1` to the entire general contact domain;
- a local assignment of `contProp2` to self-contact for `surf1`;
- a local assignment of the default Abaqus contact property to contact between `surf2` and `surf3`; and
- a local assignment of `contProp3` to contact between the entire contact domain and `surf4`. The friction coefficient for `contProp3` is reset from the initial value of 0.20 to 0.05 in the second step.

```
*SURFACE INTERACTION, NAME=contProp1
*FRICTION
0.1
*SURFACE INTERACTION, NAME=contProp2
*FRICTION
0.15
```

```
*SURFACE INTERACTION, NAME=contProp3
*FRICTION
0.20
*CONTACT
*CONTACT INCLUSIONS, ALL EXTERIOR
*CONTACT PROPERTY ASSIGNMENT
 , , contProp1
surf1, surf1, contProp2
surf2, surf3,
 , surf4, contProp3
…
*STEP
Step1
*STATIC
…
*END STEP
*STEP
Step2
*STATIC
…
*CHANGE FRICTION, INTERACTION NAME=contProp3
*FRICTION
0.05
*END STEP
```

Stabilization for General Contact in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About General Contact in Abaqus/Standard*
- **CONTACT*
- **CONTACT STABILIZATION*
- *Creating contact stabilization definitions*
- *Specifying and modifying contact stabilization assignments for general contact*

Overview

Contact stabilization for general contact in Abaqus/Standard:

- is often helpful in stabilizing unconstrained rigid body modes in static analyses;
- can be applied selectively to particular regions within a general contact domain; and
- can vary over time.

Stabilization Based on Viscous Damping of Relative Motion between Surfaces

Contact stabilization is based on viscous damping opposing incremental relative motion between nearby surfaces, in the same manner as contact damping (see *Contact Damping*). The most common purpose of contact stabilization is to stabilize otherwise unconstrained "rigid body motion" before contact closure and friction restrain such motions. A goal of artificial stabilization, such as contact stabilization, is to provide enough stabilization to enable a robust, efficient simulation without degrading the accuracy of the results. In most cases contact stabilization is not activated by default (an exception is discussed in *Contact at a Single Point*), so you will generally need to activate contact stabilization if convergence problems associated with unconstrained rigid body modes may be present in your analysis. Once activated, contact stabilization is highly automated.

The following expressions for the normal pressure, σ_{stab}, and shear stress, τ_{stab}, associated with contact stabilization involve many semiautomated factors to facilitate achieving the desired stabilization characteristics:

$$\sigma_{stab} = s_{const}\, s_{iter}\, s_{ampl}\, s_{incr}\, s_{gap}\, c_d\, v_{relN},$$

$$\tau_{stab} = s_{const}\, s_{iter}\, s_{ampl}\, s_{incr}\, s_{gap}\, s_{tang}\, c_d\, v_{relT},$$

where

c_d

is a damping coefficient;

v_{relN} and v_{relT}

are the relative normal and tangential velocities, respectively, between nearby points on opposing contact surfaces;

s_{const}

is a constant scale factor;

s_{iter}

is an iteration-dependent scale factor;

s_{ampl}

is a time-dependent scale factor;

s_{incr}

is a scale factor based on the increment number;

s_{gap}

is a scale factor based on the separation distance; and

s_{tang}

is a constant scale factor for tangential stabilization.

The damping coefficient and relative velocities are computed by Abaqus/Standard. The damping coefficient is equal to a fixed, small fraction, f, times a representative stiffness of elements underlying the contact surfaces, k_{rep}, times the time period of the step, t_{step}. Relative velocities in a static analysis are computed by dividing relative incremental displacements, Δu_{relN} and Δu_{relT}, by the time increment size, Δt.

Therefore, the following contact stabilization expressions apply to statics:

$$\sigma_{stab} = \left[s_{const} s_{iter} s_{ampl} s_{incr} s_{gap} \left(\frac{t_{step}}{\Delta t} \right) f k_{rep} \right] \Delta u_{relN},$$

$$\tau_{stab} = \left[s_{const} s_{iter} s_{ampl} s_{incr} s_{gap} s_{tang} \left(\frac{t_{step}}{\Delta t} \right) f k_{rep} \right] \Delta u_{relT},$$

where the portions within brackets can be thought of as stabilization stiffnesses (representing resistance to relative motion between nearby surfaces). The stabilization stiffness is inversely proportional to the time increment size, which is a desirable characteristic. Stabilization stiffness increases if the time increment size is reduced, which happens automatically in Abaqus/Standard if convergence difficulties occur for a particular time increment size.

Assigning Stabilization to Interactions

Contact stabilization assignments for specific interactions within general contact can be made globally or locally and are specified as part of step definitions. In most cases you only need to specify which interactions are eligible for contact stabilization without adjusting the scale factors discussed previously. Stabilization is ignored in pure heat transfer and coupled thermal-electrical procedures that do not have displacement degrees of freedom.

Input File Usage: Use the following option to specify which interactions should use contact stabilization:

**CONTACT STABILIZATION*
surf_1, *surf_2*

If the first surface name is omitted, a default surface that encompasses the entire general contact domain is assumed. If the second surface name is omitted, contact between the first surface and itself is assumed.

Abaqus/CAE Usage: Use the following options to assign contact stabilization definitions to individual surface pairs:

Interaction module: **Create Interaction**: **General contact (Standard)**: **Contact Properties**: **Stabilization assignments: Edit**: select the surfaces and the stabilization name in the columns on the left, and click the arrows in the middle to transfer them to the list of contact stabilization assignments

Specifying Stabilization Scale Factors

In some cases you may want to adjust one or more scale factors associated with contact stabilization. You can use multiple instances of this option to achieve different scale factor settings for different general contact interactions.

Constant Scale Factors

As shown in the expressions above for the stabilization pressure and shear stress, the scale factor s_{const} applies to normal and tangential stabilization, whereas the scale factor s_{tang} applies only to tangential stabilization. The default setting of the constant scale factor s_{const} is unity for the specified interactions.

The default setting of s_{tang} is zero such that no tangential stabilization stiffness exists by default for the specified interactions. Normal-direction-only contact stabilization is adequate in many cases. Other analyses can benefit from tangential stabilization stiffness; however, if you specify a nonzero setting of s_{tang}, keep in mind that tangential contact stabilization often absorbs significant energy if large relative tangential motion occurs between nearby surfaces. Large energy absorbed by stabilization is one indication that analysis results are likely to be significantly affected by the stabilization. Normal contact stabilization is much less likely to absorb significant energy and, thus, tends to have less influence on the results.

Input File Usage: **CONTACT STABILIZATION*, SCALE FACTOR=s_{const}, TANGENT FRACTION=s_{tang}

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Stabilization->Create**: **Scale factor:** s_{const}, **Tangential factor:** s_{tang}

Iteration-Dependent Scale Factors

To reduce or eliminate the likelihood of contact stabilization significantly influencing the reported solution, scale factors can be introduced that vary across iterations of an increment. Having more stabilization in effect during the early iterations of an increment can be helpful to avoid numerical problems prior to establishing active contact regions. Having less or no stabilization in effect during the later iterations can be helpful to improve the accuracy of the final converged iteration of an increment.

You can specify these scale factors. For example, specifying “1,0” results in the scale factor being unity during initial iterations (until various convergence measures are satisfied or nearly satisfied) and then the scale factor being reset to zero (effectively turning off stabilization) for the final iterations until convergence checks are again satisfied.

Input File Usage: **CONTACT STABILIZATION*, SCALE FACTOR=USER ADAPTIVE

Time-Dependent Scale Factors

The scale factors s_{ampl} and s_{incr} control time-dependence of the contact stabilization. By default, s_{ampl} is equal to the fraction of the step remaining. The other factor varies according to $s_{incr} = f_{incr}^{n_{incr}-1}$, where f_{incr} is a per-increment reduction factor (equal to 0.1 by default) and n_{incr} is the increment number within a step. These defaults imply that the stabilization is reduced by more than an order of magnitude in successive increments of the same size and that no stabilization is applied in the final increment of a step. The defaults are appropriate for most cases in which contact stabilization is intended to provide stabilization in initial increments while contact is being established.

Two options are provided for adjusting the time-dependent scale factors: you can refer to an amplitude curve that will govern s_{ampl}, and you can specify the value of f_{incr} (recall the expression $s_{incr} = f_{incr}^{n_{incr}-1}$ given previously). For example, if unstable modes remain after contact is established, you may want s_{ampl} and s_{incr} to remain equal to unity throughout a step for certain interactions, which can be accomplished by referring to an amplitude with a constant value of one and setting the per-increment reduction factor, f_{incr}, equal to one.

Input File Usage: **AMPLITUDE*, NAME=*name*
**CONTACT STABILIZATION*, AMPLITUDE=*name*,
REDUCTION PER INCREMENT=f_{incr}

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **Name:** *name*
Interaction module: **Interaction->Contact Stabilization->Create**:
Reduction factor: f_{incr}, **Amplitude:** *name*

Resetting Time-Dependent Scale Factors in Subsequent Steps

Contact stabilization definitions do not affect subsequent steps unless an amplitude reference is specified. If an amplitude based on the total time is specified, the same amplitude curve continues to govern the variation of s_{ampl} in subsequent steps until a new contact stabilization definition is assigned to the interaction. If an amplitude based on the step time is specified, the amplitude curve governs s_{ampl} for a single step and s_{ampl} remains constant (at the ending value) in subsequent steps until a new contact stabilization definition is assigned to the interaction. In both cases you can also reset the contact stabilization definition to remove stabilization from a step. Resetting ensures that contact stabilization options from prior steps do not affect the current step.

Input File Usage: **CONTACT STABILIZATION*, RESET

Abaqus/CAE Usage: Load or Interaction module: **Create Amplitude**: **Name:** *name*
Interaction module: **Interaction->Contact Stabilization->Create**:
Reset values from previous steps

Gap-Dependent Scale Factor

The scale factor s_{gap} controls contact stabilization as a function of the local separation distance between surfaces. By default, this factor is unity for zero gap distance and is zero when the gap distance is greater than or equal to a characteristic surface dimension. You can control the gap distance at which s_{gap} becomes zero. Specifying a

large value for this threshold distance is not recommended because of the tendency to increase solution cost per iteration (due to increased connectivity) as the threshold distance increases.

Input File Usage: **CONTACT STABILIZATION*, RANGE=*distance*

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Stabilization->Create**:
Zero stabilization distance: Specify: *distance*

Hierarchy of Contact Stabilization Definitions

The interface discussed above is the recommended method for specifying contact stabilization for general contact; however, contact stabilization can be introduced for general contact interactions in two other ways. The order of precedence in cases of overlap is as follows:

- First priority is given to the contact stabilization assignment options discussed in this section.
- Second priority is given to the contact stabilization assignment options discussed in *Automatic Stabilization of Rigid Body Motions in Contact Problems*.
- Third priority is given to the default contact stabilization discussed in *Contact at a Single Point*.

Differences Between Contact Stabilization Definitions

The contact stabilization definitions discussed in this section and in *Automatic Stabilization of Rigid Body Motions in Contact Problems* are similar.

Both definitions include a tangent fraction control that scales the contact stabilization coefficient in the tangential direction only. The default setting of this tangent fraction control is as follows:

- For contact stabilization discussed in this section, the default is 0.0.
- For contact stabilization discussed in *Automatic Stabilization of Rigid Body Motions in Contact Problems*, the default is 1.0.

A nonzero setting of the tangent fraction control can lead to significant energy being dissipated by contact stabilization, and a setting of zero will not stabilize relative tangential motion.

The reduction-per-increment factor is unique to the contact stabilization definition discussed in this section. The default setting of 0.1 for this factor helps enable contact to be established before the last increment of the step (when no contact stabilization effects remain by default).

Defining Contact Pairs in Abaqus/Standard

In this section:

- *About Contact Pairs in Abaqus/Standard*
- *Assigning Surface Properties for Contact Pairs in Abaqus/Standard*
- *Assigning Contact Properties for Contact Pairs in Abaqus/Standard*
- *Modeling Contact Interference Fits in Abaqus/Standard*
- *Defining Tied Contact in Abaqus/Standard*
- *Extending Main Surfaces and Slide Lines*
- *Contact Modeling If Substructures Are Present*
- *Contact Modeling If Asymmetric-Axisymmetric Elements Are Present*

About Contact Pairs in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *Element-Based Surface Definition*
- *Node-Based Surface Definition*
- *Analytical Rigid Surface Definition*
- *About Contact Interactions*
- *Contact Property Models*
- **CONTACT PAIR*
- **SURFACE*
- **MODEL CHANGE*
- *Defining surface-to-surface contact*
- *Defining self-contact*
- *Using contact and constraint detection*

Overview

Contact pairs in Abaqus/Standard:

- can be used to define interactions between bodies in mechanical, coupled temperature-displacement, coupled thermal-electrical-structural, coupled pore pressure-displacement, coupled thermal-electrical, and heat transfer simulations;
- are part of the model definition;
- can be formed using a pair of rigid or deformable surfaces or a single deformable surface;
- do not have to use surfaces with matching meshes;
- cannot be formed with one two-dimensional surface and one three-dimensional surface; and
- cannot be formed with cracked surfaces for enriched elements.

You can define contact in Abaqus/Standard in terms of two surfaces that may interact with each other as a “contact pair” or in terms of a single surface that may interact with itself in “self-contact.” Abaqus/Standard enforces contact conditions by forming equations involving groups of nearby nodes from the respective surfaces or, in the case of self-contact, from separate regions of the same surface. This section describes various aspects of defining contact pairs and refers to other sections for additional details.

Defining Contact Pairs

To define a contact pair, you must indicate which pairs of surfaces may interact with one another or which surfaces may interact with themselves. Contact surfaces should extend far enough to include all regions that may come into contact during an analysis; however, including additional surface nodes and faces that never experience contact may result in significant extra computational cost (for example, extending a secondary surface such that it includes many nodes that remain separated from the main surface throughout an analysis can significantly increase memory usage unless penalty contact enforcement is used).

Every contact pair is assigned a contact formulation (either explicitly or by default) and must refer to an interaction property. Discussion of the various available contact formulations (based on whether the tracking approach

assumes finite- or small-sliding—and whether the contact discretization is based on a node-to-surface or surface-to-surface approach) is provided in *Contact Formulations in Abaqus/Standard*. Interaction property definitions are discussed in *Assigning Contact Properties for Contact Pairs in Abaqus/Standard*.

Defining Contact between Two Separate Surfaces

When a contact pair contains two surfaces, the two surfaces are not allowed to include any of the same nodes and you must choose which surface will be the secondary and which will be the main. The selection of main and secondary surfaces is discussed in detail in *Choosing the Main and Secondary Surface Roles in a Two-Surface Contact Pair*. For simple contact pairs consisting of two deformable surfaces, the following basic guidelines can be used:

- The larger of the two surfaces should act as the main surface.
- If the surfaces are of comparable size, the surface on the stiffer body should act as the main surface.
- If the surfaces are of comparable size and stiffness, the surface with the coarser mesh should act as the main surface.

The finite-sliding, node-to-surface formulation is used by default (except in Abaqus/CAE, where the default is the finite-sliding, surface-to-surface formulation).

Defining Contact Pairs Using the Finite-Sliding, Node-to-Surface Formulation

A finite-sliding, node-to-surface formulation is available.

Input File Usage: **CONTACT PAIR*, INTERACTION=*interaction_property_name*
secondary_surface_name, *main_surface_name*

You can also specify the contact discretization directly:

**CONTACT PAIR*, INTERACTION=*interaction_property_name*,
TYPE=NODE TO SURFACE
secondary_surface_name, *main_surface_name*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface-to-surface contact (Standard)**: select the main surface, click **Surface** or **Node Region**, select the secondary surface,

Interaction editor, **Sliding formulation: Finite sliding**, **Discretization method: Node to surface**, **Contact interaction property**: *interaction_property_name*

Defining Contact Pairs Using the Finite-Sliding, Surface-to-Surface Formulation

A node-based secondary surface precludes the use of surface-to-surface discretization. Some contact capabilities are not available with the finite-sliding, surface-to-surface formulation, including crack propagation (see *Crack Propagation Analysis*).

Input File Usage: Use the following option to define contact constraints using the finite-sliding, surface-to-surface formulation:

**CONTACT PAIR*, INTERACTION=*interaction_property_name*,
TYPE=SURFACE TO SURFACE
secondary_surface_name, *main_surface_name*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface-to-surface contact (Standard)**: select the main surface, click **Surface**, select the secondary surface,
Interaction editor, **Sliding formulation: Finite sliding**, **Discretization method: Surface to surface**, **Contact interaction property**: *interaction_property_name*

Defining Contact Pairs Using the Small-Sliding, Node-to-Surface Formulation

The small-sliding tracking approach uses node-to-surface discretization by default. For an explanation of when the small-sliding tracking approach is appropriate in an analysis, see *Using the Small-Sliding Tracking Approach*.

Input File Usage: **CONTACT PAIR*, INTERACTION=*interaction_property_name*,
SMALL SLIDING
secondary_surface_name, *main_surface_name*

You can also specify the contact discretization directly:

**CONTACT PAIR*, INTERACTION=*interaction_property_name*,
SMALL SLIDING, TYPE=NODE TO SURFACE
secondary_surface_name, *main_surface_name*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface-to-surface contact (Standard)**: select the main surface, click **Surface** or **Node Region**, select the secondary surface,

Interaction editor, **Sliding formulation: Small sliding**, **Discretization method: Node to surface**, **Contact interaction property**: *interaction_property_name*

Defining Contact Pairs Using the Small-Sliding, Surface-to-Surface Formulation

A node-based secondary surface precludes the use of surface-to-surface discretization.

Input File Usage: **CONTACT PAIR*, INTERACTION=*interaction_property_name*,
SMALL SLIDING, TYPE=SURFACE TO SURFACE
secondary_surface_name, *main_surface_name*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface-to-surface contact (Standard)**: select the main surface, click **Surface**, select the secondary surface,
Interaction editor, **Sliding formulation: Small sliding**, **Discretization method: Surface to surface**, **Contact interaction property**: *interaction_property_name*

Using Symmetric Main-Secondary Contact Pairs to Improve Contact Modeling

For node-to-surface contact it is possible for main surface nodes to penetrate the secondary surface without resistance with the strict main-secondary algorithm used by Abaqus/Standard. This penetration tends to occur if the main surface is more refined than the secondary surface or a large contact pressure develops between soft bodies. Refining the secondary surface mesh often minimizes the penetration of the main surface nodes. If the refinement technique does not work or is not practical, a symmetric main-secondary method can be used if both surfaces are element-based surfaces with deformable or deformable-made-rigid parent elements. To use this method, define two contact pairs using the same two surfaces, but switch the roles of main and secondary surface for the two contact pairs. This method causes Abaqus/Standard to treat each surface as a main surface and, thus, involves additional computational expense because contact searches must be conducted twice for the same

contact pair. The increased accuracy provided by this method must be compared to the additional computational cost.

All of the contact formulations are available for symmetric main-secondary contact pairs, and can be applied using the same options discussed above.

Input File Usage: **CONTACT PAIR*, INTERACTION=*interaction_property_name*
surface_1, *surface_2*
surface_2, *surface_1*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface-to-surface contact (Standard)**: select the main surface, click **Surface**, select the secondary surface
Copy this interaction to a new interaction, and edit the new interaction. In the interaction editor, click **Switch** to reverse the main and secondary surfaces.

Limitations of Symmetric Main-Secondary Contact Pairs

Using symmetric main-secondary contact pairs can lead to overconstraint problems when very stiff or "hard" contact conditions are enforced. See *Contact Constraint Enforcement Methods in Abaqus/Standard* for a discussion of overconstraints and alternate constraint enforcement methods.

For softened contact conditions, use of symmetric main-secondary contact pairs will cause deviations from the specified pressure-versus-overclosure behavior, because both contact pairs contribute to the overall interface stress without accounting for one another. For example, symmetric main-secondary contact pairs effectively double the overall contact stiffness if a linear pressure-overclosure relationship is specified.

Likewise, use of symmetric main-secondary contact pairs will cause deviations from the friction model if an optional shear stress limit is specified (see *Using the Optional Shear Stress Limit*), because the contact stresses observed by each contact pair will be approximately one-half of the total interface stress.

Similarly, it can be difficult to interpret the results at the interface for symmetric main-secondary contact pairs. In this case both surfaces at the interface act as secondary surfaces, so each has contact constraint values associated with it. The constraint values that represent contact pressures are not independent of each other. Therefore, the constraint values reported in the data (`.dat`) and results (`.fil`) files represent only a part of the total interface pressure and have to be summed to obtain the total.

In the output database, mechanical contact variables are reported at the nodes on both the main and secondary surfaces per contact pair and not just the secondary surface where constraints are formed. Consequently, two result sets are available per surface of a symmetric main-secondary contact pair; once when a surface acts as a secondary and once as a main. For nodal contact pressures the Visualization module of Abaqus/CAE only reports the maximum of the two pressure values associated with a node when the surface containing the node acts either as a main or as a secondary surface. Even in this case, the contact pressures do not represent the true interface pressure.

Apart from contact pressures, some contact output may be confusing with symmetric main-secondary contact pairs. For example, Abaqus/Standard may report a positive opening distance on one side of a contact interface but zero opening distance (i.e., touching) on the opposite side of the interface. Typically this is caused by the shape or relative mesh refinement of the two surfaces.

Defining Self-Contact

Define contact between a single surface and itself by specifying only a single surface or by specifying the same surface twice. The small-sliding tracking approach cannot be used with self-contact.

Defining Self-Contact Using Node-to-Surface Discretization

Abaqus/Standard uses node-to-surface contact discretization by default for self-contact.

Input File Usage: Use either of the following options:

**CONTACT PAIR*, INTERACTION=*interaction_property_name*
surface_1,

**CONTACT PAIR*, INTERACTION=*interaction_property_name*
surface_1, *surface_1*

Abaqus/CAE Usage: Interaction module: **Create Interaction**:
Self-contact (Standard): select the surface
Interaction editor, **Discretization method: Node to surface**, **Contact interaction property**: *interaction_property_name*

or

Interaction module: **Create Interaction**: **Surface-to-surface contact (Standard)**: select the surface, click **Surface**, select the surface again
Interaction editor, **Sliding formulation: Finite sliding**, **Discretization method: Node to surface**, **Contact interaction property**: *interaction_property_name*

Defining Self-Contact Using Surface-to-Surface Discretization

Surface-to-surface discretization often leads to more accurate modeling of self-contact simulations. However, because the self-contact surface is acting as both a main and a secondary, surface-to-surface discretization can sometimes significantly increase the solution cost.

Input File Usage: Use either of the following options:

**CONTACT PAIR*, INTERACTION=*interaction_property_name*,
TYPE=SURFACE TO SURFACE
surface_1,

**CONTACT PAIR*, INTERACTION=*interaction_property_name*,
TYPE=SURFACE TO SURFACE
surface_1, *surface_1*

Abaqus/CAE Usage: Interaction module: **Create Interaction**:
Self-contact (Standard): select the surface
Interaction editor, **Discretization method: Surface to surface**, **Contact interaction property**: *interaction_property_name*

or

Interaction module: **Create Interaction**: **Surface-to-surface contact (Standard)**: select the surface, click **Surface**, select the surface again
Interaction editor, **Sliding formulation: Finite sliding**, **Discretization method: Surface to surface**, **Contact interaction property**: *interaction_property_name*

Limitations of Self-Contact

Self-contact is valid only for mechanical surface interactions and is limited to finite sliding with element-based surfaces.

A node of a self-contact surface can be both a secondary node and a member of the main surface for two-dimensional self-contact using the surface-to-surface formulation and for all three-dimensional self-contact. In these cases the contact behavior is similar to symmetric main-secondary contact pairs, and the issues discussed in *Using Symmetric Main-Secondary Contact Pairs to Improve Contact Modeling* apply. Abaqus/Standard automatically applies some numerical "softening" to contact conditions in these cases, as discussed in *Contact Constraint Enforcement Methods in Abaqus/Standard*.

Direct enforcement of hard contact conditions is the default constraint enforcement method for two-dimensional self-contact using the node-to-surface formulation. In this case, each node adjacent to a vertex where a two-dimensional surface folds onto itself is automatically assigned a secondary or main role during the analysis. Since contact constraints directly resist penetrations at nodes that act as secondary nodes, there is some possibility of unresolved penetrations at nodes that only act as main nodes for two-dimensional self-contact using the node-to-surface formulation.

Selecting Surfaces Used in Contact Pairs

Methods for creating surfaces are discussed in *Element-Based Surface Definition*, *Node-Based Surface Definition*, and *Analytical Rigid Surface Definition*; those sections discuss general restrictions for the various surface types. Considerations related to surface characteristics for various contact formulations are discussed in *Contact Formulations in Abaqus/Standard*. Additional considerations for surfaces used in contact definitions are discussed below.

Orientation Considerations for Shell-Like Surfaces

Abaqus/Standard requires main contact surfaces to be single-sided for node-to-surface contact and for some surface-to-surface contact formulations (see *Fundamental Choices Affecting the Contact Formulation* for details). This requires that you consider the proper orientation for main surfaces defined on elements, such as shells and membranes, that have positive and negative directions. For node-to-surface contact the orientation of secondary surface normals is irrelevant, but for surface-to-surface contact the orientation of single-sided secondary surfaces is taken into consideration.

Double-sided element-based surfaces are allowed for the default surface-to-surface contact formulations, although they are not always appropriate for cases with deep initial penetrations. If the main and secondary surfaces are both double-sided, the positive or negative orientation of the contact normal direction will be chosen such as to minimize (or avoid) penetrations for each contact constraint. If either or both of the surfaces are single-sided, the positive or negative orientation of the contact normal direction will be determined from the single-sided surface normals rather than the relative positions of the surfaces.

When the orientation of a contact surface is relevant to the contact formulation, you must consider the following aspects for surfaces on structural (beam and shell), membrane, truss, or rigid elements:

- Adjacent surface faces must have consistent normal directions. Abaqus/Standard will issue an error message if adjacent surface faces have inconsistent normals on a single-sided surface whose orientation is relevant to the contact formulation.
- Except for initial interference fit problems (see *Modeling Contact Interference Fits in Abaqus/Standard*), the secondary surface should be on the same side of the main surface as the outward normal. If, in the initial configuration, the secondary surface is on the opposite side of the main surface as the outward normal, Abaqus/Standard will detect overclosure of the surfaces and may have difficulty finding an initial solution if the overclosure is severe. An improper specification of the outward normal will often cause an analysis to immediately fail to converge. *Figure 1* illustrates the proper and improper specification of a main surface's outward normal.

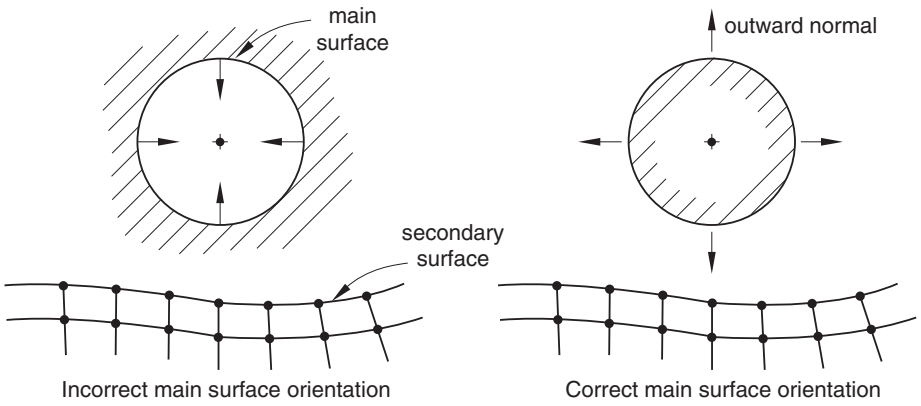


Figure 1: Example of proper and improper main surface orientation.

- Contact will be ignored with surface-to-surface discretization if single-sided secondary and main surfaces have normal directions that are in approximately the same direction (for example, contact will not be enforced if the dot product of the secondary and main surface normals is positive).

The following output from a data check analysis (see *Abaqus/Standard and Abaqus/Explicit Execution*) can be useful in identifying incorrectly oriented main surfaces:

- Initial clearances can be displayed in Abaqus/CAE with a contour plot of the variable COPEN at increment 0 of the first step; initial overclosures correspond to negative clearances.
- Abaqus/Standard provides a detailed printout of the model's initial contact state.

Surface Connectivity Restrictions

Certain connectivity restrictions apply to contact surfaces depending on the type of contact formulation. Surface connectivity restrictions for the various contact formulations are summarized in *Table 1*. As indicated in this table, the connectivity restrictions are sometimes different for main and secondary surfaces. Self-contact surfaces act as both main and secondary surfaces; therefore, if a restriction applies to either a main or secondary surface, it also applies to self-contact. The potential connectivity restrictions referred to in *Table 1* are described below:

Table 1: Summary of which connectivity characteristics of element-based surfaces are allowed for various contact formulations.

Contact formulation	Connectivity characteristics	
	Discontinuous (or 3D faces joined at only one node)	T-intersection
Finite-sliding, node-to-surface	Main: Not allowed Secondary: Allowed	Main: Not allowed Secondary: Allowed
Small-sliding, node-to-surface	Main: Allowed Secondary: Allowed	Main: Not allowed Secondary: Allowed
Finite-sliding, surface-to-surface	Main: Allowed Secondary: Allowed	Main: Allowed Secondary: Allowed
Small-sliding, surface-to-surface	Main: Allowed Secondary: Allowed	Main: Allowed Secondary: Allowed

- Discontinuous surfaces: Discontinuous contact surfaces are allowed in many cases, but the main surface for finite-sliding, node-to-surface contact cannot be made up of two or more disconnected regions (they must be continuous across element edges in three-dimensional models or across nodes in two-dimensional models).

Figure 2 shows examples of continuous surfaces, whereas *Figure 3* and *Figure 4* show examples of discontinuous surfaces. *Figure 5* shows an automatically generated free surface resulting from the specification of an element set consisting of two disjointed groups of elements. The resulting surface is not continuous since it is composed of two disjoint open curves, so this surface would be invalid as a main surface for finite-sliding, node-to-surface contact.

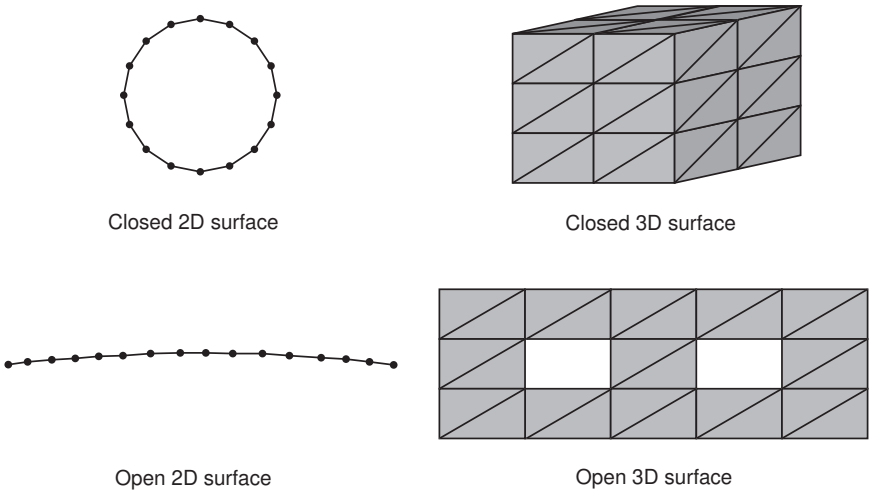


Figure 2: Examples of continuous surfaces.



Figure 3: Example of a discontinuous 2D surface.

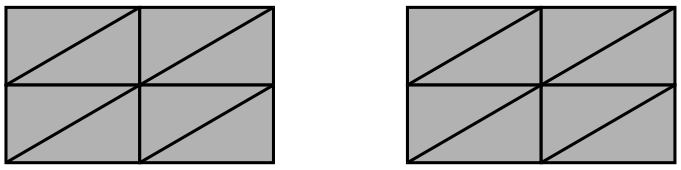


Figure 4: Example of a discontinuous 3D surface.

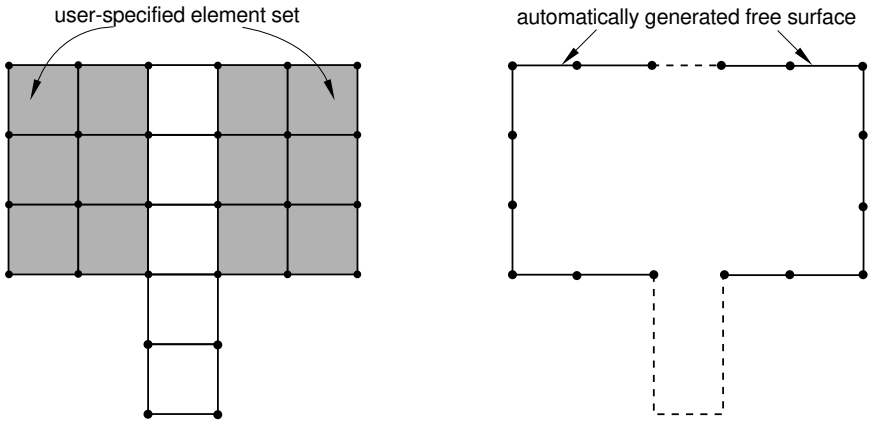


Figure 5: Example of a discontinuous surface resulting from automatic free surface generation with a disjoint element set.

- Portions of three-dimensional surfaces joined at only one node: The finite-sliding, node-to-surface contact formulation also does not allow three-dimensional main surface faces to be joined at a single node (they must

be joined across a common element edge). *Figure 6* shows an example of a surface with two faces connected by a single node.

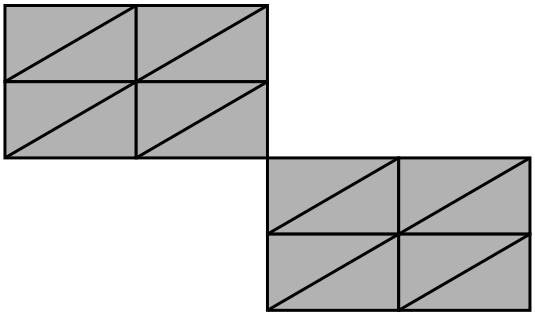


Figure 6: Example of a 3D surface with two faces sharing a single node.

- Surfaces with T-intersections: In some cases a contact surface cannot have more than two surface faces sharing a common main node in two dimensions or a common main edge in three dimensions. For example, *Figure 7* shows examples of surfaces with T-intersections, in which three faces share a common node in two dimensions or a common edge in three dimensions. While more than two surface faces can share a common secondary node in two dimensions or a common edge in three dimensions for node-to-surface formulations, the secondary faces must be single-sided, which precludes the most common T-intersection cases for node-to-surface formulations.

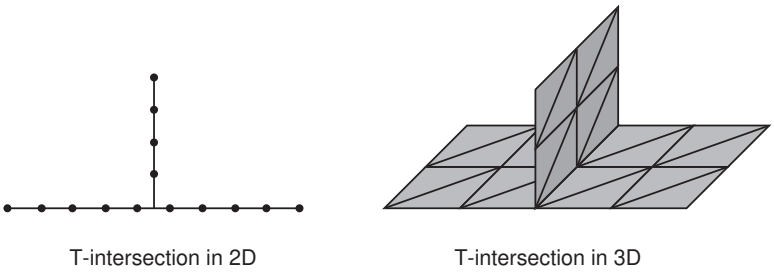


Figure 7: Examples of surfaces with T-intersections.

Analytical Rigid Surfaces

Analytical rigid surfaces are often effective for efficiently modeling curved, rigid geometries, as discussed in *Analytical Rigid Surface Definition*. For rare cases in which a very large number (thousands) of segments would be necessary to define an analytical rigid surface, better performance can be achieved with an element-based rigid surface (see *Element-Based Surface Definition*).

Three-Dimensional Beam and Truss Surfaces

Abaqus/Standard cannot use three-dimensional beams or trusses to form a main surface because the elements do not have enough information to create unique surface normals. However, these elements can be used to define a secondary surface. Two-dimensional beams and trusses can be used to form both main and secondary surfaces.

Beam section offsets, specified either as part of the beam cross section definition or through the **BEAM SECTION OFFSET* option, are neglected for contact interactions in Abaqus/Standard.

Edge-Based Surfaces

Edge-based surfaces (*Element-Based Surface Definition*) on three-dimensional shell elements cannot be used in a contact analysis in Abaqus/Standard.

Limitations of Node-Based Surfaces

Use node-based surfaces with caution when the contact property definition includes user-defined softened contact properties or thermal or electrical interactions because the contact constitutive behavior (which relies on accurate calculation of contact pressure, heat flux, or electric current) will not be enforced correctly unless the precise surface area is associated with each node. For details, see *Contact Pressure-Overclosure Relationships*, *Thermal Contact Properties*, or *Electrical Contact Properties*.

Removing and Reactivating Contact Pairs

You can temporarily remove contact pairs from a simulation, which may result in significant computational savings by eliminating unnecessary contact searches and updates of surface orientations during the simulation. Removal and reactivation of contact pairs is commonly used in complicated forming processes where multiple tools need to interact with the workpiece at different stages in the analysis. Step-dependent activation and deactivation are also supported for general contact when used as an alternative to contact pairs for modeling contact (see *Deactivating and Reactivating Contact Interactions across Steps*).

You cannot remove tied contact pairs from a simulation (see *Defining Tied Contact in Abaqus/Standard*).

Removing Contact Pairs

Removal of contact pairs is a useful technique for uncoupling components of an assembly until they should be brought together (such as tooling in manufacturing process simulations). Significant computational expense might be saved by removing a contact pair and introducing it at the proper time, thus eliminating the need to monitor the contact conditions except when they are relevant.

Input File Usage: **MODEL CHANGE*, TYPE=CONTACT PAIR, REMOVE
secondary_surface, main_surface

Repeat the data line as needed.

Abaqus/CAE Usage: Use one of the following options:

Interaction module: **Create Interaction**: surface-to-surface contact or self-contact interaction editor: toggle off **Active in this step**

Interaction module: interaction manager: select interaction, **Deactivate**

Removal of Contact Forces Associated with Closed Contact Pairs

If the surfaces are in contact when a contact pair is removed, Abaqus/Standard stores the corresponding contact forces (or heat fluxes if thermal interactions are present, or electrical currents if it is a coupled-thermal electrical analysis) for every node on each surface. Abaqus/Standard automatically ramps these forces (or heat fluxes or electrical currents) linearly down to zero magnitude during the removal step. Abaqus/Standard always removes the contact constraints for mechanical surface interactions instantaneously.

Care must be taken in removing contact pairs in transient procedures. In transient heat transfer, fully coupled temperature-displacement, or fully coupled thermal-electrical-structural analysis if the fluxes are high and the step is long, this ramping down may have the effect of cooling down or heating up the rest of the body. In dynamic analysis if the forces are high and the step is long, kinetic energy can be imparted to the remaining portion of the model. This problem can be avoided by removing the contact pairs in a very short transient step prior to the rest of the analysis. This step can be done in a single increment.

Using an Allowable Contact Interference to Deactivate Contact Pairs

A contact pair with mechanical contact interactions can be deactivated during an analysis by assigning a very large allowable contact interference to the contact pairs (see *Modeling Contact Interference Fits in Abaqus/Standard*). This method has the disadvantage of not reducing the computational cost of the analysis because the contact algorithm will still calculate the contact conditions for the contact pair in each increment.

Reactivating Contact Pairs

All contact pairs that will be used in a simulation must be created at the start of the analysis; they cannot be created once the simulation has begun. However, contact pairs can be created, removed at the start of the analysis in the first step, and then reactivated at a later point during the simulation.

In Abaqus/CAE you can create contact pairs in any step. If a contact pair is created in a step other than the initial step, Abaqus/CAE automatically deactivates the contact pair in the initial step and reactivates it in the step in which you created it.

Input File Usage: **MODEL CHANGE*, TYPE=CONTACT PAIR, ADD
secondary_surface, main_surface

Repeat the data line as needed.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: surface-to-surface contact or self-contact
interaction editor: toggle on **Active in this step**

Reactivating Overclosed Contact Pairs

When a contact pair is reactivated, the contact constraint becomes active immediately. In mechanical simulations it is possible for the surfaces of a contact pair to move such that they become overclosed while the contact pair is inactive. If this overclosure is too severe when the contact pair is reactivated, Abaqus/Standard may encounter convergence problems as it tries to enforce the suddenly activated contact constraint. To avoid such problems, you can specify a permissible interference value, v, for the contact pair that is larger than the overclosure for the contact pair. Abaqus/Standard will ramp v down to zero during the step. For details on specifying allowable interferences, see *Modeling Contact Interference Fits in Abaqus/Standard*.

Output

Output variables associated with the interaction of contact pairs fall into two categories: nodal variables (sometimes called constraint variables) and whole surface variables. In addition, Abaqus outputs an array of diagnostic information associated with contact interactions, as discussed in *Contact Diagnostics in an Abaqus/Standard Analysis*.

For more detailed discussions of variables associated with thermal, electrical, and pore fluid analyses, see the sections on the related contact properties in *Contact Property Models*.

Nodal Contact Variables

Nodal contact variables can be contoured on contact surfaces in the Visualization module of Abaqus/CAE. Nodal contact variables include contact pressure and force, frictional shear stress and force, relative tangential motion (slip) of the surfaces during contact, clearance between surfaces, heat or fluid flux per unit area, fluid pressure, and electrical current per unit area. Many of the nodal contact variables written to the output database (`.odb`) file are often available for all contact nodes, regardless of whether they act as secondary or main nodes. Other nodal contact variables are available only at nodes acting as secondary nodes. Most contact output to the data (`.dat`) file, results (`.fil`) file, and the utility subroutine `GETVRMAVGATNODE` is associated with individual

constraints. For contact output to the output database (`.odb`) file, some filtering is applied to reduce contact output noise.

The contact pressure distribution is of key interest in many Abaqus analyses. You can view the contact pressure on all contact surfaces except for analytical rigid surfaces and discrete rigid surfaces based on rigid-type elements (the latter restriction does not apply to general contact). You can view a contour plot of the contact pressure error indicator next to a contour plot of the contact pressure to gain perspective on local accuracy of the contact pressure solution in regions where the contact pressure solution is of interest (see *Selection of Error Indicators Influencing Adaptive Remeshing* for further discussion of error indicator output).

In some cases you may observe the contact pressure extending beyond the actual contact zone due to the following factors:

- The contour plots are constructed by interpolating nodal values, which can cause nonzero values to appear within portions of facets outside of the contact region. For example, this effect is often noticeable at corners, such as when two same-sized, aligned blocks are in contact—if the contact surfaces wrap around the corners, the contact pressure contours will extend slightly around the corners.
- Abaqus/Standard outputs postprocessed contact stresses to the output database. During postprocessing, nodal contact stresses are calculated as weighted averages of values associated with active contact constraints in which the node participates. For example, a main node can participate in multiple constraints whose connectivities contain the node. Similarly, a secondary node can participate in multiple constraints, as in the case of the finite sliding surface-to-surface formulation when more than one constraint is created at secondary node locations that are corners and edge features.

  The weighting depends on the contact constraint area and a scaling factor based on the strength of the participation of the node in a constraint. The weighted averaging is intended to reduce contact stress noise. Modifications are made for calculating weightings, for example, at the corner nodes of quadratic faces (with zero consistent nodal areas) and main nodes that are outside the active contact region but participate in contact weakly through an active contact constraint. These modifications also have a filtering effect in terms of reducing contact stress values reported for nodes on the fringe of the active contact region. For such locations, contact nodal areas that are simply cumulative scaled constraint areas across constraints in which a node participates do not have much bearing on contact stress values.

  In addition to averaging and filtering, contact stresses are also smoothed during the postprocessing operations. However, this filtering and subsequent smoothing are not "perfect" and can result in the contact zone size appearing somewhat exaggerated. Similarly, contact status output is also affected at nodes that lie on the fringe of the active contact region. In such cases, the contact status may be reported as closed at nodes in the exaggerated region even though it is open.

Due to these factors, trying to infer the contact force distribution from the contact stress distribution can be somewhat misleading. Instead, you can request nodal contact force output, which accurately represents the contact force distribution present in the analysis.

Whole Surface Variables

Whole surface variables are attributes of an entire secondary surface (see *Mechanical Analysis–Whole Surface Quantities*). Available as history output, these variables record the total force and moment due to contact pressure and frictional stress, the center of pressure and frictional stress (defined as the point closest to the centroid of the surface that lies on the line of action of the resultant force for which the resultant moment is minimal), and the total contact area (defined as the sum of all the facets where there is contact force). The last letter of each variable name (except the variable CAREA) denotes which contact force distribution on the surface is used to calculate the resultant:

N

Normal contact forces are used to derive the resultant quantity.

S

Shear contact forces are used to derive the resultant quantity.

T

The sum of the normal and shear contact forces is used to derive the resultant quantity.

For example, CFN is the total force due to contact pressure, CFS is the total force due to frictional stress, and CFT is the total force due to both contact pressure and frictional stress.

Each total moment output variable will not necessarily equal the cross product of the respective center of force vector and resultant force vector. Forces acting on two different nodes of a surface may have components acting in opposite directions, such that these nodal force components generate a net moment but not a net force; therefore, the total moment may not arise entirely from the resultant force. The center of force output variables tend to be most meaningful when the surface nodal forces act in approximately the same direction.

Requesting Output

Certain contact variables must be requested as a group. For example, to output the clearance between surfaces (COPEN), you must request the variable CDISP (contact displacements). CDISP outputs both COPEN and CSLIP (tangential motion of the surfaces during contact). A complete listing of available contact pair variables and identifiers is given in *Abaqus/Standard Output Variable Identifiers*.

Output requests can be limited to individual contact pairs or portions of a secondary surface. You can:

- request output associated with a given contact pair;
- request output associated with a given secondary surface, including contributions from all of the contact pairs to which the secondary surface belongs; and
- limit the output by specifying a node set containing a subset of the nodes on the secondary surface.

Instructions on forming these output requests are available in the following sections:

- To request output to the data (`.dat`) file, see *Surface Output from Abaqus/Standard*.
- To request output to the output database (`.odb`) file, see *Writing Surface Output to the Output Database*.

Differences for Small-Sliding and Finite-Sliding Contact

For small-sliding contact problems the contact area is calculated in the input file preprocessor from the undeformed shape of the model; thus, it does not change throughout the analysis, and contact pressures for small-sliding contact are calculated according to this invariant contact area. This behavior is different from that in finite-sliding contact problems, where the contact area and contact pressures are calculated according to the deformed shape of the model.

Output of Tangential Results

Abaqus reports the values of tangential variables (frictional shear stress, viscous shear stress, and relative tangential motion) with respect to the local tangent directions defined on the surfaces. The local tangent directions CTANDIR1 and CTANDIR2 can be output by requesting the generic output variable CTANDIR. The definition of local tangent directions is explained in *Local Tangent Directions on a Surface*. These directions do not always correspond to the global coordinate system, and they rotate with the contact pair in a geometrically nonlinear analysis.

Abaqus/Standard calculates tangential results at each constraint point by taking the scalar product of the variable's vector and a local tangent direction, $\mathbf{t}_1$ or $\mathbf{t}_2$, associated with the constraint point. The number at the end of a variable's name indicates whether the variable corresponds to the first or second local tangent direction. For example, CSHEAR1 is the frictional shear stress component in the first local tangent direction, while CSHEAR2 is the frictional shear stress component in the second local tangent direction.

Definition of Accumulated Incremental Relative Motion (Slip)

Abaqus/Standard defines the incremental relative motion (also known as slip) as the scalar product of the incremental relative nodal displacement vector and a local tangent direction. The incremental relative nodal displacement vector measures the motion of a secondary node relative to the motion of the main surface. The incremental slip is accumulated only when the secondary node is contacting the main surface. The sums of all such incremental slips during the analysis are reported as CSLIP1 and CSLIP2. Details about the calculation of this quantity can be found in *Small-sliding interaction between bodies*, *Finite-sliding interaction between deformable bodies*, and *Finite-sliding interaction between a deformable and a rigid body*.

Extending the Range for Which Contact Opening Output Is Provided for Gaps

To reduce computational costs, detailed computations to monitor potential points of interaction are avoided by default where surfaces are separated by a distance greater than the minimum gap distance at which contact forces (or thermal fluxes, etc.) may be transmitted. Therefore, contact opening (COPEN) output is typically not provided for finite-sliding contact where surfaces are opened by more than a small amount compared to surface facet dimensions. You can extend the range in which Abaqus/Standard provides contact opening output; COPEN will be provided up to gap distances equal to a specified "tracking thickness." Using this control may increase computational cost due to extra contact tracking computations, especially if you specify a large tracking thickness value.

Input File Usage: **SURFACE INTERACTION*, TRACKING THICKNESS=*value*

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Geometric Properties**: toggle on **Thickness to determine the contacting surfaces to be tracked:** *value*

Output for Axisymmetric Models

In an axisymmetric analysis the total forces and moments transmitted between the contacting bodies as a result of contact pressure and frictional stress are computed in the same manner as in a two-dimensional analysis. Therefore, the component of the total forces along the *r*-axis is nonzero, and the components of the total moments include contributions from the total forces along the *r*-axis.

Obtaining the "Maximum Torque" That Can Be Transmitted about the Z-Axis in an Axisymmetric Analysis

When modeling surface-based contact with axisymmetric elements (element types CAX and CGAX), Abaqus/Standard can calculate the maximum torque (output variable CTRQ) that can be transmitted about the *z*-axis. This capability is often of interest when modeling threaded connectors (see *Axisymmetric analysis of a threaded connection*). The maximum torque, T, is defined as

$$T = \int\int r^2 p \, ds \, d\theta,$$

where p is the pressure transmitted across the interface, r is the radius to a point on the interface, and s is the current distance along the interface in the r–z plane. This definition of "torque" effectively assumes a friction coefficient of unity.

Whole Model Contact-Related Energy Variables

For contact pairs, the same contact-related energy variables are available in Abaqus/Standard as for general contact, as described in *Whole Model Contact-Related Energy Variables*.

Assigning Surface Properties for Contact Pairs in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Contact Pairs in Abaqus/Standard*
- **CONTACT PAIR*
- *Defining surface-to-surface contact*
- *Defining self-contact*

Overview

This section describes how to modify the properties associated with surfaces in a contact pair definition.

Accounting for Shell and Membrane Thickness

All of the contact formulations except the finite-sliding, node-to-surface formulation account for initial shell and membrane thicknesses for element-based surfaces by default. The finite-sliding, node-to-surface formulation will not account for surface thickness. Node-based surfaces have no thickness, regardless of which element types are connected to the surface nodes. Accounting for element thicknesses in contact calculations is generally desirable, but you can avoid having thickness considered if it is not desired.

Input File Usage: **CONTACT PAIR*, NO THICKNESS

Abaqus/CAE Usage: Interaction module: interaction editor: **Sliding formulation: Small sliding** or **Finite sliding**, **Discretization method**: **Surface to surface** or **Node to surface**, toggle on **Exclude shell/membrane element thickness**

Example

Consider the case of a shell pinched between two rigid surfaces, as shown in *Figure 1*.

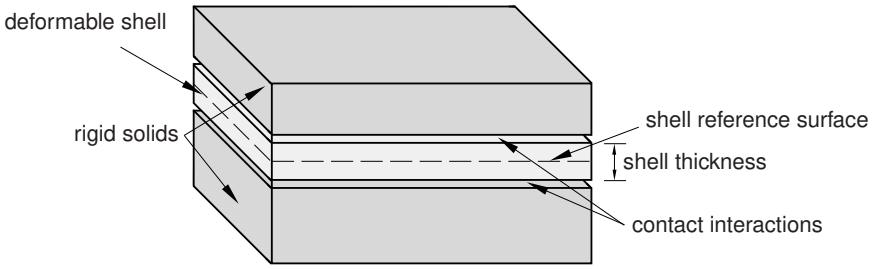


Figure 1: Shell pinched between two rigid bodies.

In this example contact pairs using the small-sliding, node-to-surface formulation are defined between the top surface of the shell and the top rigid surface and between the bottom surface of the shell and the bottom rigid surface. Although the shell surfaces are defined at the shell reference location, the contact interactions account for the thickness of the shell and are offset from the reference surface. The penalty constraint enforcement method

(see *Contact Pressure-Overclosure Relationships*) is used to avoid overconstraining secondary nodes. The following input is used:

```
*SURFACE, NAME=TOP_RIG_SURF
TOP_RIG_ELS,
*SURFACE, NAME=SHELL_TOP_SURF
SHELL_ELS,SPOS
*SURFACE, NAME=SHELL_BOT_SURF
SHELL_ELS,SNEG
*SURFACE, NAME=BOT_RIG_SURF
BOT_RIG_ELS,
*CONTACT PAIR, INTERACTION=INTER_AL, SMALL SLIDING
SHELL_TOP_SURF, TOP_RIG_SURF
SHELL_BOT_SURF, BOT_RIG_SURF
*SURFACE INTERACTION, NAME=INTER_AL
*SURFACE BEHAVIOR, PENALTY
```

Specifying Surface Geometry Corrections

With the finite element method, curved geometric surfaces are naturally approximated as a faceted group of connected element faces. The use of a faceted surface geometry rather than the true surface geometry can significantly contribute to contact stress inaccuracy in contact pairs, especially when the magnitude of the differences between the faceted and true surface is not small with respect to the deformation of the components in contact. Methods for overcoming convergence and accuracy difficulties associated with faceted surfaces in contact interactions are discussed in *Contact Formulations in Abaqus/Standard* and *Smoothing Contact Surfaces in Abaqus/Standard*.

Assigning Contact Properties for Contact Pairs in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Contact Interactions*
- *About Contact Pairs in Abaqus/Standard*
- **CONTACT PAIR*
- **SURFACE INTERACTION*
- *Defining surface-to-surface contact*
- *Defining self-contact*
- *Using contact and constraint detection*

Overview

Contact properties:

- define the mechanical and thermal surface interaction models that govern the behavior of surfaces when they are in contact; and
- are assigned to individual contact pairs.

Assigning a Surface Interaction Definition to a Contact Pair

A surface interaction definition specifies the constitutive contact properties and the constraint enforcement methods used by a contact pair. Every contact pair in a model must refer to a surface interaction definition, even if the contact pair uses the default contact property models. See *About Mechanical Contact Properties* for information on defining contact properties. A non-default constraint enforcement method can be specified as part of a surface interaction definition, as described in *Contact Constraint Enforcement Methods in Abaqus/Standard*.

Multiple contact pairs can refer to the same surface interaction definition.

The surface interaction definition is also used to provide a non-default value for the surface out-of-plane thickness in the following cases:

- a two-dimensional model, or
- a contact pair involving a node-based surface to provide a non-default cross-sectional area at every node in the node-based surface.

Abaqus/Standard uses the out-of-plane surface thickness and cross-sectional area provided in the surface interaction definition to compute the contact stresses.

Input File Usage: Use both of the following options:

**CONTACT PAIR*, INTERACTION=*interaction_property_name*
**SURFACE INTERACTION*, NAME=*interaction_property_name*

Abaqus/CAE Usage: Interaction module:

Create Interaction Property: **Name:** *interaction_property_name*, **Contact**

Interaction editor: **Contact interaction property**: *interaction_property_name*

Example

Figure 1 shows the mesh used in this example. For purposes of this example, the surface `ASURF` is the secondary surface of the contact pair. The property definition for the contact pair (`GRATING`) uses the finite-sliding, node-to-surface formulation with a friction model with μ=0.4 and uses the default "hard" contact model for the behavior normal to the surfaces.

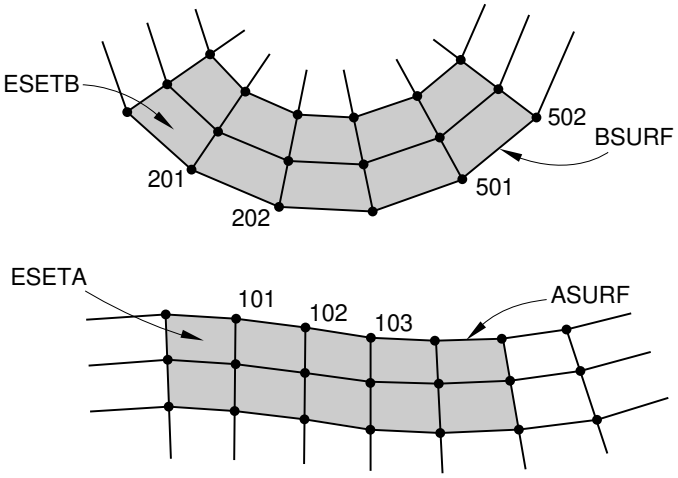


Figure 1: Mechanical surface interaction with friction and finite sliding.

```
*HEADING
…
*SURFACE, NAME=ASURF
ESETA,
*SURFACE, NAME=BSURF
ESETB,
*CONTACT PAIR, INTERACTION=GRATING
ASURF, BSURF
*SURFACE INTERACTION, NAME=GRATING
*FRICTION
0.4
*NSET, NSET=SNODES
101, 102, 103
*STEP, NLGEOM
…
*END STEP
```

Modeling Contact Interference Fits in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Contact Pairs in Abaqus/Standard*
- **CONTACT INTERFERENCE*
- *Specifying interference fit options*

Overview

Interference fits in Abaqus/Standard:

- occur by default when the contact formulation computes overclosures between surfaces in the initial configuration of a model;
- are resolved in the first increment of a step by default;
- can be gradually resolved over multiple increments;
- result in stresses and strains in a model as overclosures are resolved;
- can be specified for both surface-based contact pairs and contact elements; and
- cannot be specified for self-contact.

Abaqus/Standard offers alternative methods to resolve initial overclosures with strain-free adjustments and to model specific overclosures or clearances different from those calculated from the initial configuration. These methods are discussed in *Contact Initialization for Contact Pairs in Abaqus/Standard*.

Resolving Excessive Initial Overclosures

If there are large overclosures in the initial configuration of model, Abaqus/Standard may not be able to resolve the interference fit in a single increment. Abaqus/Standard provides alternative methods that allow overclosures to be resolved gradually over multiple increments.

The default contact constraint imposed at each constraint location is that the current penetration $h(t)$ is ≤ 0.

Penetration exists when $h(t)$ is positive. To alter this constraint, you can specify an allowable interference, v, that will be ramped down over the course of a step. The specified allowable interference modifies the contact constraint as follows:

$$h(t) - v(t) \leq 0.$$

Thus, specifying a positive value for $v(t)$ causes Abaqus/Standard to ignore penetrations up to that magnitude. *Figure 1* illustrates a typical interference fit problem.

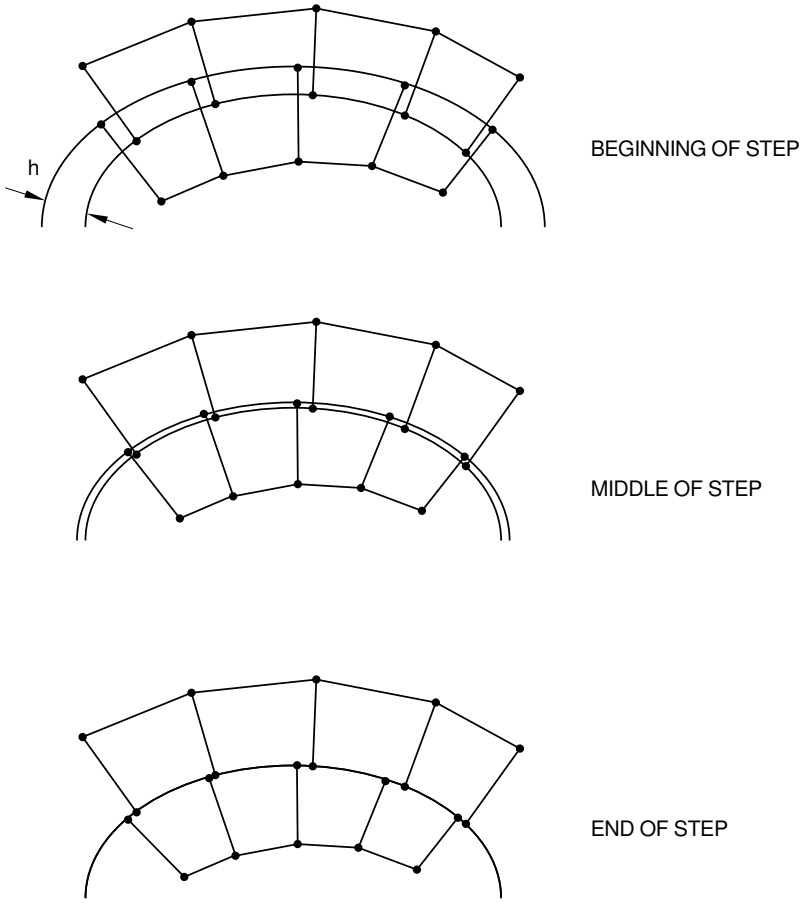


Figure 1: Interference fit with contact surfaces.

If the penetration in the model is h, you may declare $v = h$ or request an automatic shrink fit. In either case Abaqus/Standard will consider the two bodies to be just in contact at the start of the simulation. As the allowable interference, v, is decreased during the step, Abaqus/Standard pushes the surfaces apart until there is no more allowable penetration.

There are three different ways in which to specify the allowable interference, v. By default, in all cases the value of the specified allowable interference is applied instantaneously at the start of the step and then ramped down to zero linearly over the step, unless you specify an amplitude reference that defines a particular allowable interference-time variation. It is recommended that you specify allowable interferences in a step separate from the rest of the analysis; additional loads may adversely affect the resolution of the interference fit and the response to loading with partially-resolved interferences may be non-physical. Once the overclosures are resolved, you can continue the analysis in a new step.

When the contact interference is specified, output variable COPEN does not reflect the actual overclosure value during the step; it reflects the actual value only at the end of the step.

You must specify the contact pairs or contact elements at which the allowable interference should apply.

Input File Usage: Use the following option to define an allowable interference for contact pairs:

**CONTACT INTERFERENCE*, TYPE=CONTACT PAIR
secondary surface, *main surface*, v
...

Use the following option to define an allowable interference for contact elements:

**CONTACT INTERFERENCE*, TYPE=ELEMENT
contact element set, v
...

Abaqus/CAE Usage: Interaction module: interaction editor: **Interference Fit**: **Gradually remove secondary node overclosure during the step**, **Uniform allowable interference**, **Magnitude at start of step:** v

Element-based contact is not supported in Abaqus/CAE.

Using a Nondefault Amplitude Curve for the Allowable Interference

You can define a time-varying allowable contact interference by creating an amplitude curve (see *Amplitude Curves* for details) and then referring to this curve from the contact interference definition. The amplitude will be ignored, however, if the Riks method (see *Unstable Collapse and Postbuckling Analysis*) is used.

Input File Usage: **CONTACT INTERFERENCE*, AMPLITUDE=*amplitude_curve_name*

Abaqus/CAE Usage: Interaction module: interaction editor: **Interference Fit**: **Gradually remove secondary node overclosure during the step**, **Uniform allowable interference**, **Amplitude:** *amplitude_curve_name*

Removing or Modifying the Allowable Contact Interferences

By default, only the allowable contact interferences defined or redefined by a particular contact interference definition will be modified. Alternatively, you can specify that all previously defined allowable contact interferences should be removed from the model and only those defined with this definition will remain.

Input File Usage: Use the following option to add or modify an allowable contact interference definition:

**CONTACT INTERFERENCE*, OP=MOD

Use the following option to remove all previously defined allowable contact interferences:

**CONTACT INTERFERENCE*, OP=NEW

Abaqus/CAE Usage: Contact interferences in Abaqus/CAE propagate along with the interaction for which they are defined. You cannot remove all previously defined contact interferences at once in Abaqus/CAE.

Specifying the Same Allowable Contact Interference for an Entire Surface

A single allowable interference v can be specified for every node on the secondary surface or every secondary node in the specified set of contact elements. The concepts of secondary nodes for the various families of contact elements are discussed in their respective sections. The specified allowable contact interferences are included in the current penetrations of the secondary nodes reported in the message file when you request detailed contact printout. Thus, any secondary node that penetrates the main surface by less than the allowable interference will be reported as being open.

Using the Automatic “Shrink” Fit Method

This method is applicable only during the first step of an analysis and requires no interference value. With this method Abaqus/Standard assigns a different v to each secondary node that is equal to that node's initial penetration (or zero if the point is initially open) except for the finite-sliding, surface-to-surface formulation, in which case the same value of v, corresponding to the maximum penetration of the contact pair, is assigned to all constraints that are initially closed. These automatically calculated allowable contact interferences are not included in the current penetrations reported in the message file when detailed contact printout is requested.

When the automatic “shrink” fit method is used, only the default amplitude curve, a linear ramp to zero magnitude, can be used.

Input File Usage: **CONTACT INTERFERENCE*, SHRINK

Abaqus/CAE Usage: Interaction module: interaction editor: **Interference Fit**: **Gradually remove secondary node overclosure during the step**, **Automatic shrink fit**

Applying an Allowable Contact Interference with a Shift Vector

In this method you specify a uniform allowable interference v and a direction $\mathbf{n}$. The allowable interference value, v, defines the magnitude of a shift vector. A relative shift $v\mathbf{n}$ is applied to the secondary nodes before Abaqus/Standard determines the contact conditions. In certain applications, such as contact simulations of threaded connectors, shifting the surfaces in a specified direction is more effective than simply allowing an interference.

Figure 2 illustrates the potential difference that can result when using an allowable contact interference with a shift vector rather than using a uniform allowable contact interference. In case (a) a shift direction $\mathbf{n}$ is defined as well as an allowable interference v, while in case (b) the standard approach is used, with an allowable interference v.

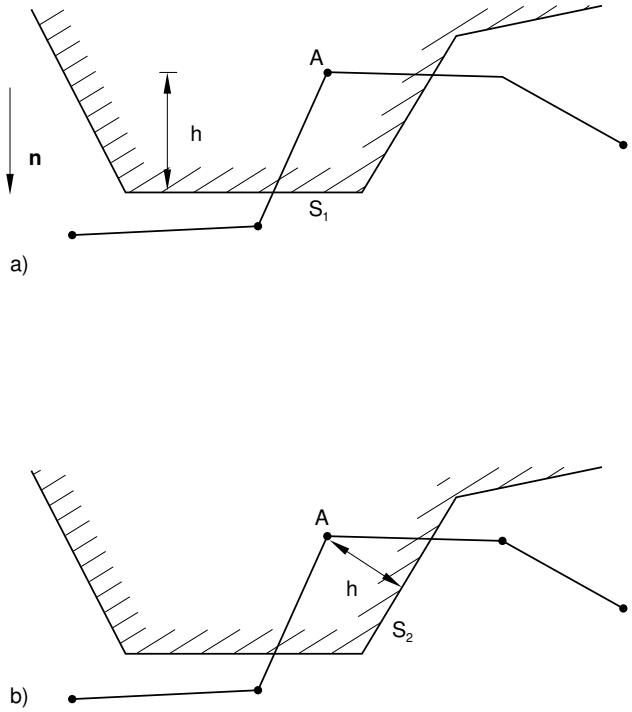


Figure 2: Effect of direction definition on interference accommodation: a) with direction, b) without direction.

The magnitude of v is the same in both cases, but it is less than the penetration in case (a) and more than the penetration in case (b). In case (a) contact is detected immediately for secondary node *A*, and the penetration is resolved with that node sliding along segment s_1 because node *A* is shifted in the direction $\mathbf{n}$ before Abaqus/Standard checks for contact. After the shift Abaqus/Standard determines that node *A* is closest to segment s_1 and moves the node onto that segment. In case (b) secondary node *A* detects contact with segment s_2 because that is the closest segment when node *A* remains in its initial position. Thus, node *A* will slide along segment s_2 if no shift direction is provided.

Input File Usage: **CONTACT INTERFERENCE*
secondary surface, *main surface*, v, *X-direction cosine of* $\mathbf{n}$, *Y-direction cosine of* $\mathbf{n}$, *Z-direction cosine of* $\mathbf{n}$
...

Abaqus/CAE Usage: Interaction module: interaction editor: **Interference Fit**: **Gradually remove secondary node overclosure during the step**, **Uniform allowable interference**, **Magnitude at start of step:** v, **Along direction:** X, Y, Z

Interference Fits for Surface-to-Surface Discretization

Because contact conditions are enforced in an average sense in a region around each constraint location for surface-to-surface contact, penetrations or gaps may be observed at secondary nodes when surface-to-surface constraints are in a zero-penetration state.

Large interferences may be difficult to resolve with the finite-sliding, surface-to-surface formulation. Using this formulation, overclosures tend to be resolved along the secondary facet normal directions; using node-to-surface contact, overclosures tend to be resolved along the main surface normal directions. *Figure 3* illustrates a case

where differing normal directions lead to undesirable tangential motion during an interference fit. In some cases it may be preferable to resolve large initial overclosures with node-to-surface discretization.

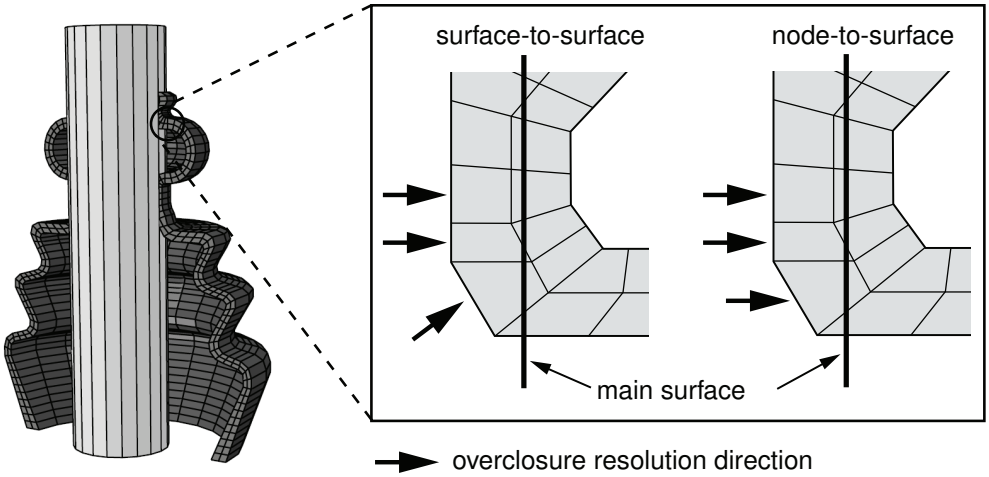


Figure 3: Comparison of contact formulations in an example with a large interference fit.

Friction and Contact Interferences

Frequently, an actual assembly process is modeled as an interference fit problem. If frictional interface properties are desired, they should usually be introduced after the initial interference has been resolved. The initial interference problem should be modeled under frictionless conditions since the physical assembly process is not typically modeled exactly. Friction can be introduced in subsequent steps (see *Changing Friction Properties during an Abaqus/Standard Analysis*).

Defining Tied Contact in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Contact Pairs in Abaqus/Standard*
- *Contact Initialization for Contact Pairs in Abaqus/Standard*
- **CONTACT PAIR*
- *Defining surface-to-surface contact*
- *Using contact and constraint detection*

Overview

Tied contact in Abaqus/Standard:

- ties two surfaces forming a contact pair together for the duration of a simulation;
- can be used in mechanical, coupled temperature-displacement, coupled thermal-electrical-structural, coupled pore pressure-displacement, coupled thermal-electrical, or heat transfer simulations;
- constrains each of the nodes on the secondary surface to have the same value of displacement, temperature, pore pressure, or electrical potential as the point on the main surface that it contacts;
- allows for rapid transitions in mesh density within the model;
- requires a nondefault setting to control the treatment of small initial overclosures and gaps; and
- cannot be used with self-contact or symmetric main-secondary contact.

It is often preferable to use the surface-based tie constraint capability instead of tied contact (see *Mesh Tie Constraints* for details).

Defining Tied Contact for a Contact Pair

To "tie" the surfaces of a contact pair together for an analysis, you must specify a nondefault control such that small initial overclosures (and optionally small initial gaps) are resolved by either adjusting the surface positions automatically or storing a contact offset distance per tied secondary node such that the initial penetration is zero. See *Contact Initialization for Contact Pairs in Abaqus/Standard* for details on adjusting surfaces. As always, you must associate the contact pair with a contact interaction property definition.

Input File Usage: Use one of the following options to adjust surface positions as necessary for tied regions, where *a* is the threshold initial gap distance that is tied:

**CONTACT PAIR*, TIED, ADJUST=*a* or *node_set_label*, INTERACTION=*name*

**CONTACT PAIR*, TIED, ADJUST=YES, POSITION TOLERANCE=*a*, INTERACTION=*name*

Use the following option to store an offset distance (instead of adjusting surface positions) as necessary for tied regions, where *a* is the threshold initial gap distance that is tied:

*CONTACT PAIR, TIED, ADJUST=NO, POSITION TOLERANCE=*a*, INTERACTION=*name*

Abaqus/CAE Usage: Interaction module: **Interaction->Create**: select a **Secondary Node/Surface Adjustment** option: toggle on **Tie adjusted surfaces**

Storing penetration offsets in the contact penetration calculations is not supported in Abaqus/CAE.

The Tied Contact Formulation

When a contact pair uses the tied contact formulation, Abaqus/Standard uses the undeformed configuration of the model to determine which secondary nodes are within the adjustment zone (see *Adjusting Initial Surface Positions to Resolve Small Initial Gaps or Overclosures*), accounting for any shell or membrane thickness by default. Abaqus/Standard then either adjusts these secondary nodes' positions or determines a contact offset distance such that the revised initial penetration distance is zero. It forms constraints between these secondary nodes and the surrounding nodes on the main surface, using either a "surface-to-surface" or a "node-to-surface" approach, similar to small-sliding contact. The traditional node-to-surface approach is used by default for tied contact.

The user interface for selecting between the surface-to-surface and node-to-surface approaches and to avoid consideration of shell and membrane thickness for tied contact is the same as for small-sliding contact (see *About Contact Pairs in Abaqus/Standard* and *Assigning Surface Properties for Contact Pairs in Abaqus/Standard*).

Use of Tied Contact in Mechanical Simulations

The tied contact formulation constrains only translational degrees of freedom in mechanical simulations. Abaqus/Standard places no constraints on the rotational degrees of freedom of structural elements involved in tied contact pairs.

Self-contact is not supported with tied contact. Self-contact is designed for finite-sliding situations in which it is not obvious from the original geometry which parts of the surface will come into contact during the deformation.

Mechanical constraints for tied contact are strictly enforced with a direct Lagrange multiplier method by default. Alternatively, you can specify that these constraints should be enforced with a penalty or augmented Lagrange constraint method (see *Contact Constraint Enforcement Methods in Abaqus/Standard*). The constraint enforcement method specified will be applied to the tangential constraints in addition to the normal constraints. Softened contact pressure-overclosure relationships (exponential, tabular, or linear—see *Contact Pressure-Overclosure Relationships*) are ignored for tied contact.

Use of Tied Contact in Nonmechanical Simulations

The tied contact capability can be used in models where the nodal degrees of freedom include electrical potential and/or temperature. Except for the nodal degree of freedom being constrained, Abaqus/Standard uses exactly the same formulation for tied contact in nonmechanical simulations as it does for mechanical simulations.

Unconstrained Nodes in Tied Contact Pairs

Abaqus/Standard does not constrain secondary nodes to the main surface unless they are precisely in contact with the main surface at the start of the analysis. Any secondary nodes not precisely in contact at the start of the analysis—e.g., either open or overclosed—will remain unconstrained for the duration of the simulation; they will never interact with the main surface. In mechanical simulations an unconstrained secondary node can penetrate the main surface freely. In a thermal, electrical, or pore pressure simulation an unconstrained secondary node will not exchange heat, electrical current, or pore fluid with the main surface.

To avoid such unconstrained nodes in tied contact pairs, use the capability for adjusting the surfaces of a contact pair described in *Contact Initialization for Contact Pairs in Abaqus/Standard*. This capability moves secondary nodes onto the main surface before Abaqus/Standard checks for the initial contact state. It is intended only for nodes that are close to the main surface and is not intended to correct large errors in the mesh geometry.

Checking That Secondary Nodes Are Constrained

Abaqus/Standard prints a table in the data (`.dat`) file identifying the predominant secondary node and other nodes involved in each constraint. If Abaqus/Standard cannot form a constraint for a given secondary node acting as a predominant secondary node, either because it is not in contact with the main surface or it cannot "see" the main surface, it will issue a warning message in the data file. For an explanation of when a secondary node would not "see" a main surface and how to correct this problem, see *Contact Formulations in Abaqus/Standard*. When creating a model with tied contact, it is important to use this information provided by Abaqus/Standard to identify any unconstrained nodes and to make any necessary modifications to the model to constrain them.

Extending Main Surfaces and Slide Lines

Products: Abaqus/Standard

References:

- *About Contact Pairs in Abaqus/Standard*
- *Common Difficulties Associated with Contact Modeling in Abaqus/Standard*
- **CONTACT PAIR*
- **SLIDE LINE*

Overview

Extending the main surface or a slide line:

- can prevent nodes from “falling off” or getting trapped behind the main surface (or slide line) in finite-sliding problems;
- allows the secondary node to find a main surface when the secondary node has no intersection with the main surface at the start of the analysis in small- and infinitesimal-sliding problems;
- can avoid numerical roundoff difficulties associated with contact modeling;
- should not be used in lieu of proper contact modeling techniques;
- should not be used to reduce the number of underlying elements of a contact surface;
- applies only at the perimeter of a main surface in three dimensions and at the ends of a main surface in two dimensions; and
- applies only to contact pairs that use a node-to-surface discretization.

Extending the Main Surface for Small-Sliding, Node-to-Surface Contact

If a secondary node cannot find an intersection with the main surface at the start of the analysis, it will be free to penetrate the main surface because no local tangent plane will be formed. This type of problem, which typically occurs for node-to-surface contact when the secondary node is aligned with the end or perimeter of the main surface (which does not wrap around the corner of the rectangular body), is illustrated in *Figure 1* and may be caused by numerical roundoff errors when a preprocessor is used to generate the nodal coordinates. There are no extensions to main faces in the interior of a surface. If the main surface in *Figure 1* were defined such that it wrapped around the corner of the body, no extensions to the main surface would be required because the secondary node would project onto the main surface using the projection method discussed in *Using the Small-Sliding Tracking Approach*. Cases such as that shown in *Figure 1* are not problematic for the small-sliding, surface-to-surface formulation because the constraint formulation considers the region of the secondary surface near a secondary node.

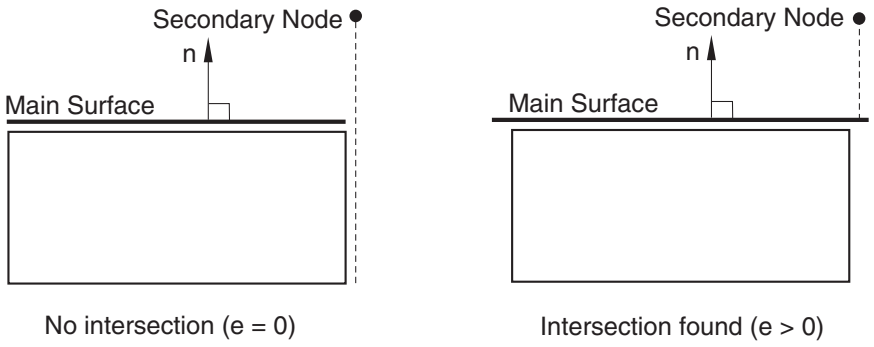


Figure 1: Secondary node fails to find an intersection with the main surface for small-sliding, node-to-surface contact if *e=0*.

For node-to-surface contact you can specify the size of the extension zone, *e*, as a fraction of the end segment or facet edge length (see *Figure 2*). If *e* is set to zero, Abaqus will not extend the ends. The value given must lie between 0.0 and 0.2. The default value is 0.1 for node-to-surface contact; surface extensions are not available for surface-to-surface contact.

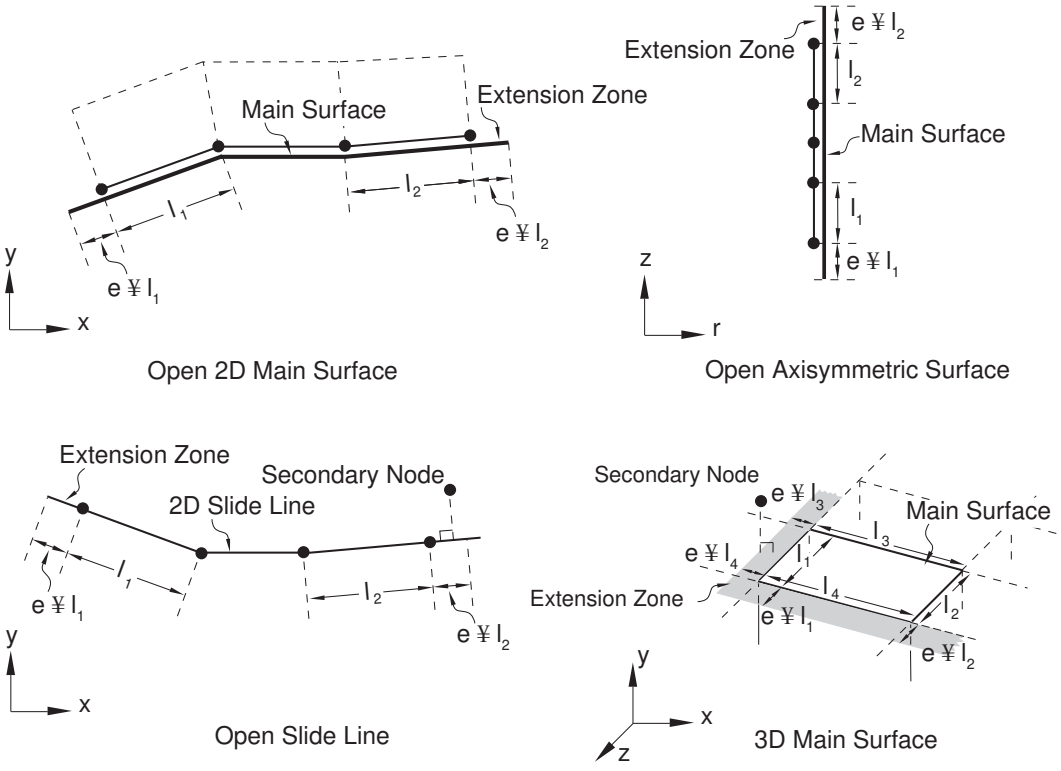


Figure 2: Definition of size of extension zone.

Input File Usage: **CONTACT PAIR*, SMALL SLIDING, EXTENSION ZONE=*e*

Extending the Main Surface or Slide Line in Finite-Sliding, Node-to-Surface Contact

To prevent secondary nodes from “falling off” or getting trapped behind a main surface, an open surface or slide line can be extended beyond its perimeter edges (in three dimensions) or end nodes (in two dimensions) for finite-sliding, node-to-surface contact.

You can specify the size of the extension zone, *e*, as a fraction of the end segment or facet edge length (see *Figure 2*). The geometry in the extension zone is extrapolated from the end segment or facet edge. If *e* is set to zero, Abaqus/Standard will not extend the ends. The value given must lie between 0.0 and 0.2. The default value is 0.1 for node-to-surface contact. Surface extensions are not available for surface-to-surface contact; for finite-sliding, surface-to-surface contact, constraints are located within secondary faces, and "falling off" will not occur until nearly the entire secondary facet slides off the main surface. Extensions for finite-sliding, node-to-surface contact should be considered only if other modeling techniques to prevent "falling off" are not feasible and when the secondary node is expected to travel in the extended zone for a short period of the solution phase or during nonconverged iterations.

Input File Usage: Use either of the following options:

**CONTACT PAIR*, EXTENSION ZONE=*e*
**SLIDE LINE*, ELSET=*element_set_name*, EXTENSION ZONE=*e*

Contact Modeling If Substructures Are Present

Products: Abaqus/Standard

References:

- *Element-Based Surface Definition*
- *Node-Based Surface Definition*
- *Using Substructures*
- *Membrane Elements*
- *Surface Elements*
- *About Contact Interactions*
- *About Contact Pairs in Abaqus/Standard*

Overview

Contact in Abaqus/Standard involving substructures:

- is not part of the substructure definition;
- requires retaining nodes on the exterior of the substructure;
- requires the definition of a contact surface on the retained nodes; and
- can be between the exterior of one substructure and another surface, the exterior of one substructure and the exterior of another substructure, and the exterior of one substructure and itself.

Defining the Contact Surface of a Substructure

Since a substructure consists only of a group of retained nodal degrees of freedom, it has no surface geometry upon which Abaqus/Standard can define a contact surface. One of the following methods must be used to define the surface geometry of the substructure:

- mesh the exterior of the substructure with surface elements,
- mesh the exterior of the substructure with structural elements,
- use a node-based surface, or
- use contact elements.

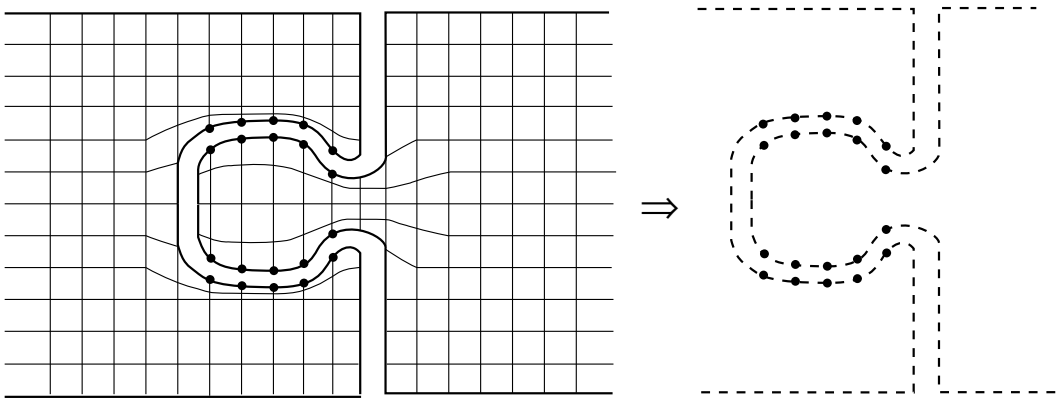
Meshing the surface of the substructure with surface or structural elements provides the most flexibility in defining the contact conditions; the surface can be used as either a main or secondary surface in the simulation. Using a node-based surface is probably the easiest method to use, but the limitations inherent to node-based surfaces (such as the inability to act as a main surface, the need to define nodal contact areas for exact contact stress recovery, and the lack of visualization of contact stresses) may limit the usefulness of this approach. Contact elements can be a useful method if the model uses matched meshes.

Meshing the Surface of the Substructure with Surface Elements

The surface geometry of the body being modeled with a substructure can be designated by defining elements on the retained surface nodes of the substructure. The elements can be used to create an element-based surface (see *Element-Based Surface Definition*), which can then be used as part of a contact pair.

Whenever possible, it is recommended that you use surface elements to mesh the exterior of a substructure. Surface elements will accurately define the surface geometry of the substructure without introducing any additional stiffness to the model; the stiffness of the underlying body is built into the substructure. See *Surface Elements* for more information about surface elements.

Figure 1 shows a simulation where both of the contacting bodies have been modeled with substructures. The nodes retained in the model are indicated in the figure. If this were a three-dimensional model, general surface elements would be used to reconstruct the appropriate surface geometries of the original mesh.



(a) critical model

(b) nodes retained for contact resolution

Figure 1: Substructuring in a contact simulation.

Limitations of Surface Elements

Surface elements cannot be used to overlay substructures in planar models.

Surface elements also cannot be used to overlay a substructure that consists of second-order, three-dimensional elements with midface nodes (C3D27(R)(H) or C3D15V(H)). Surface elements with midface nodes are not currently available in Abaqus/Standard, and the 8-node surface element (SFM3D8) is not well suited for contact modeling.

Meshing the Surface of the Substructure with Structural Elements

Although surface elements are generally preferable for use in substructure contact situations, you can also use structural elements to define the surface geometry of a substructure. You can use membrane elements in three-dimensional models and axisymmetric models, and trusses in planar models. Define the elements to have very small thickness or area and define their material property to have a very small elastic modulus so that their contribution to the stiffness of the model is negligible.

If the model in *Figure 1* were a planar model, truss elements would be used to connect the nodes and define the surface geometry. The truss elements would have a very small cross-sectional area and refer to a material property with very low stiffness so that they do not add any significant stiffness to the underlying bodies.

Limitations of Structural Elements

Membrane elements cannot be used to overlay a substructure that consists of second-order, three-dimensional brick elements of type C3D20(R)(H) if the substructure will be used as a secondary surface. Normally, Abaqus/Standard automatically converts C3D20(R)(H) brick elements to elements with midface nodes C3D27(R)(H) because this class of elements performs better in contact simulations. Abaqus/Standard also converts any second-order, three-dimensional structural element that does not have a midface node when it is used in a secondary surface (see *Three-Dimensional Surfaces with Second-Order Faces and a Node-to-Surface*

Formulation for details). Therefore, if second-order membrane elements (type M3D8) are used to reconstruct the surface topology of a substructure consisting of C3D20 elements, Abaqus/Standard will convert them to M3D9 elements when the surface is used as a secondary surface. The midface nodes that are generated automatically will not correspond to any retained nodes and, thus, will have zero stiffness. The lack of stiffness at these nodes will cause numerical problems during the analysis. Membrane elements can be used if elements of type C3D27(R)(H) have been used on the surface of the substructure.

Using a Node-Based Surface to Define the Substructure's Surface

If the retained nodes of the substructures are associated with the secondary surface of a contact pair, the retained nodes can be included in a node-based surface (see *Node-Based Surface Definition*). In this case it is not necessary to overlay the surface of the substructure with elements.

Using Contact Elements to Define the Substructure's Surface

GAP elements (*Gap Contact Elements*) can be used to define the contact interactions in the model. These elements require that matching nodes be present on the opposite sides of the contact surfaces and allow only for small relative sliding between the surfaces. This latter assumption is usually consistent with the assumption of linear behavior that is built into a substructure.

Contact Modeling If Asymmetric-Axisymmetric Elements Are Present

Products: Abaqus/Standard

References:

- *Slide Line Contact Elements*
- *Rigid Surface Contact Elements*
- **ASYMMETRIC-AXISYMMETRIC*

Overview

Modeling contact in asymmetric-axisymmetric problems:

- requires the use of contact elements (ISL or IRS);
- requires independent contact elements on each circumferential plane; and
- can be done only on certain circumferential planes.

Modeling Contact in Asymmetric-Axisymmetric Problems

CAXA or SAXA elements (see *Axisymmetric Solid Elements with Nonlinear, Asymmetric Deformation* and *Axisymmetric Shell Elements with Nonlinear, Asymmetric Deformation*) are used to model problems where initially axisymmetric structures may undergo asymmetric deformations. These asymmetric deformations may include asymmetric contact conditions. The surface-based contact capability cannot be used to model such problems; contact elements (ISL or IRS) must be used.

Independent sets of two-dimensional contact elements must be created for each circumferential plane in the CAXA or SAXA elements. You must specify the angle, θ, of the circumferential plane with which each set of contact elements is associated and the number of Fourier modes, n, used with the underlying CAXA or SAXA elements.

Input File Usage: Use both of the following options:

**INTERFACE*, ELSET=*element_set_name*
**ASYMMETRIC-AXISYMMETRIC*, MODE=n, ANGLE=θ

where the ELSET parameter refers to a set of ISL- or IRS-type contact elements.

Limitations on Contact in Asymmetric-Axisymmetric Problems

If the circumferential planes in an asymmetric-axisymmetric problem rotate more than a few degrees, Abaqus/Standard can model contact conditions correctly only on the θ=0 and 180 circumferential planes. The asymmetric-axisymmetric elements have internal degrees of freedom for the rotation and out-of-plane motion of the circumferential planes, but these degrees of freedom are not accounted for in the contact elements. Ignoring these degrees of freedom means that Abaqus/Standard keeps the contact directions fixed in initial circumferential planes and the position of the nodes is projected back onto these initial planes for contact calculations. If the rotation and motion of the nodes from these initial planes are small, the errors caused by this approach are minimal. If they are large, the errors will become very large, making the results unrealistic.

Defining General Contact in Abaqus/Explicit

In this section:

- *About General Contact in Abaqus/Explicit*
- *Assigning Surface Properties for General Contact in Abaqus/Explicit*
- *Assigning Contact Properties for General Contact in Abaqus/Explicit*

About General Contact in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *About Contact Interactions*
- **CONTACT*
- **CONTACT INCLUSIONS*
- **CONTACT EXCLUSIONS*
- *Defining general contact*

Overview

Abaqus/Explicit provides two algorithms for modeling contact and interaction problems: the general contact algorithm and the contact pair algorithm. See *About Contact Interactions* for a comparison of the two algorithms. This section describes how to include general contact in an Abaqus/Explicit analysis, how to specify the regions of the model that might be involved in general contact interactions, and how to obtain output from a general contact analysis.

The general contact algorithm in Abaqus/Explicit:

- is specified as part of the model or history definition of the model;
- allows very simple definitions of contact with very few restrictions on the types of surfaces involved;
- uses sophisticated tracking algorithms to ensure that proper contact conditions are enforced efficiently;
- can be used simultaneously with the contact pair algorithm (that is, some interactions can be modeled with the general contact algorithm, while others are modeled with the contact pair algorithm);
- can be used with two-dimensional, axisymmetric, and three-dimensional surfaces;
- can be used only in mechanical finite-sliding contact analyses; and
- does not support kinematic constraint enforcement (contact constraints are enforced with the penalty method).

Defining a General Contact Interaction

The definition of a general contact interaction consists of specifying:

- the general contact algorithm and defining the contact domain (that is, the surfaces that interact with one another), as described in this section;
- the contact surface properties (*Assigning Surface Properties for General Contact in Abaqus/Explicit*);
- the mechanical contact property models (*Assigning Contact Properties for General Contact in Abaqus/Explicit*);
- the contact formulation (*Contact Formulation for General Contact in Abaqus/Explicit*);
- the initial clearance between contact surfaces (*Contact Initialization for General Contact in Abaqus/Explicit*); and
- the algorithmic contact controls (*Contact Controls for General Contact in Abaqus/Explicit*).

Surfaces Used for General Contact

The general contact algorithm allows for very general characteristics in the surfaces that it uses, as discussed in *About Contact Interactions*. For detailed information on defining surfaces in Abaqus/Explicit for use with the general contact algorithm, see *Element-Based Surface Definition*, *Node-Based Surface Definition*, *Analytical Rigid Surface Definition*, *Eulerian Surface Definition*, and *Operating on Surfaces*.

In addition, Abaqus/Explicit automatically defines an all-inclusive surface that is convenient for prescribing the contact domain, as discussed later in this section. The all-inclusive automatically defined surface includes all element-based surface facets as well as all analytical rigid surfaces and surfaces on all Eulerian materials.

The general contact algorithm generates contact forces to resist node-into-face, node-into-analytical rigid surface, and edge-into-edge contact penetrations. *Figure 1* shows a contact domain involving solid, shell, and beam elements. The primary mechanism for enforcing contact is node-to-face contact (the only mechanism used in the contact pair algorithm). If analytical rigid surfaces are present in the contact domain, the general contact algorithm also enforces node-to-analytical rigid surface contact.

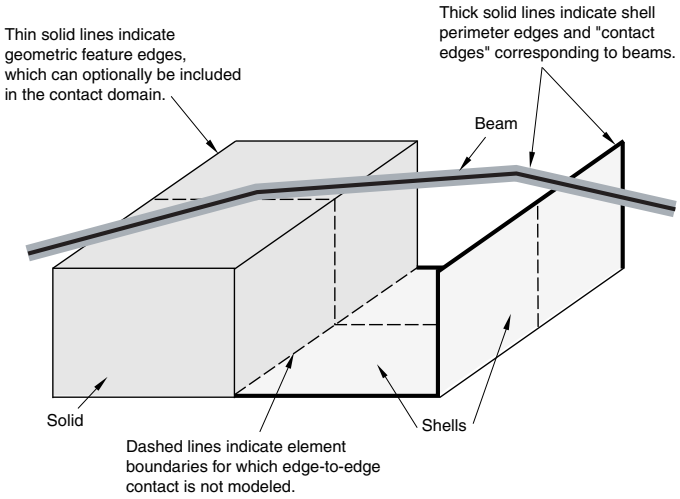


Figure 1: An example general contact domain involving solid, shell, and beam elements.

General contact can be used for a two-dimensional (plane strain, plane stress, or axisymmetric) model, in which case node-into-face and node-into-analytical rigid surface contact is considered.

For three-dimensional models general contact algorithm also considers edge-to-edge contact, which is very effective in enforcing contact that cannot be detected as penetrations of nodes into faces. The terminology “contact edges” refers to feature edges of surface facets (on both shells and solids) as well as to segments representing beam and truss elements. For example, contact between beam segments and shell perimeter edges (see *Figure 1*) is usually detected as edge-to-edge contact. Edge-to-edge contact often resists penetrations involving feature edges. By default, when a surface is used in a general contact interaction, all applicable facets, analytical rigid surfaces, nodes, perimeter edges, currently active feature edges, and beam and truss segments are included in the contact definition. You can control which feature edges are considered for edge-to-edge contact, as discussed in *Assigning Surface Properties for General Contact in Abaqus/Explicit*.

Contact edges associated with shell perimeters and shell feature edges represent the shell thickness in the normal direction and do not extend past the perimeter (similar to shell nodes and facets). Some numerical rounding of features occurs for both node-to-facet and edge-to-edge contact. See *Contact Surface Representation for Beams and Trusses* for a discussion of contact surface geometry for these element types.

Contact Surface Representation for Beams and Trusses

General contact considers realistic beam cross-section geometry in contact calculations by default. *Figure 2* shows a rendering of a variety of beam cross-sections. For beams with circular cross-sections, general contact uses contact edges and contact nodes with the same radius as the beam cross-section. Truss cross-sections are always treated as circular for contact calculations. By default, general contact uses internally generated meshes of quadrilateral surface elements to represent the contact surface geometry of beams with noncircular cross-sections (such as shown in *Figure 3*) for beams meshed with B31 and B32 elements having box cross-sections, respectively.

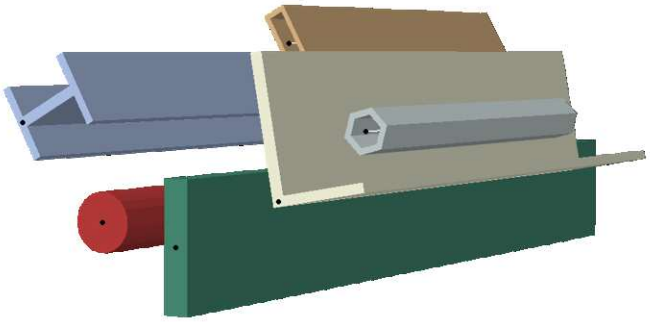


Figure 2: Beam element rendering showing various cross-sections.

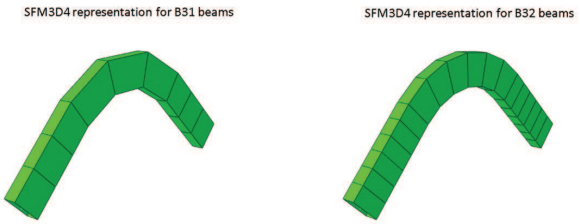


Figure 3: Surface element representation for beams with a box cross-section.

Some rounding of beam edges in the contact representation (which is beneficial for contact numerics) is achieved by offsetting the nodal positions of the internally generated surface-element meshes inward from the vertex locations of the cross-section at the axial locations of the beam nodes. General contact introduces a corresponding surface thickness automatically such that the contact representation is as shown in *Figure 4* for a rectangular cross-section. Although the surface thickness is not represented on surface elements during postprocessing, this thickness is accounted for in the contact calculations. Therefore, the contact surface representation appears slightly smaller during postprocessing than the actual beam cross-section.

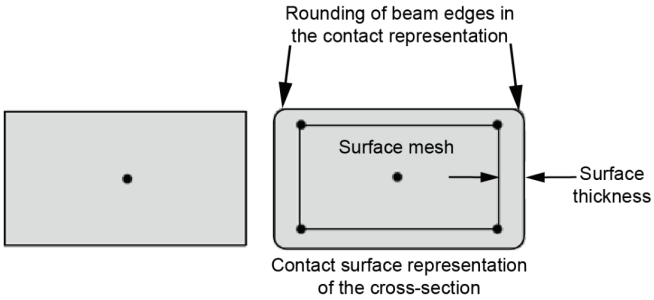


Figure 4: Contact surface representation of a rectangular beam cross-section.

Directly assigning a nonzero surface thickness for contact (see *Surface Thickness*) to a beam with a noncircular cross-section will contribute additional contact surface thickness to the contact representation of the beam, as shown in *Figure 5*.

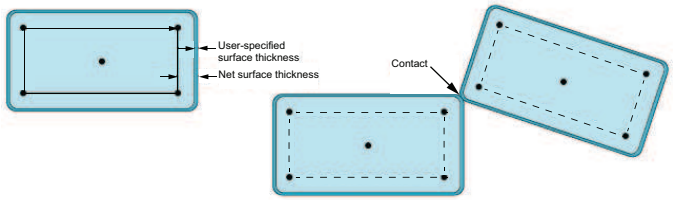


Figure 5: Effect of user-specified surface thickness for a rectangular beam cross-section.

Motion of each surface-element node is driven by the corresponding beam node via an internally generated beam MPC (see *Using MPC Type BEAM*). Abaqus redistributes the contact forces acting on these nodes to the original beam nodes as forces and moments to provide the correct beam response. The surface element-based surfaces inherit the associated general contact property assignments from the corresponding beam surfaces. Meshes of quadrilateral surface elements for noncircular beam cross-sections are available for postprocessing (see *Output for Beams with Actual Beam Cross-Section Representation*).

A beam represented by reference-line edges and nodes of beam elements for contact can participate in general contact in Abaqus/Explicit in the following ways:

- Nodes of circular beams can participate in node-to-surface contact.
- Edges of circular beams can participate in edge-to-edge contact.

A noncircular beam represented by internally generated surface elements for contact can participate in general contact in Abaqus/Explicit in the following ways:

- Nodes and faces of the surface element representation can participate in node-to-surface contact.
- Edges of the surface element representation can participate in edge-to-edge contact.

Alternatively, the contact surface geometry of a noncircular beam cross-section can be approximated with a circular cross-section encompassing the actual cross-section. *Figure 6* shows rendering of beam elements with a rectangular cross-section on the left side and the optional, nondefault circular representation for contact on the right side. The nodal contact area assigned to the beam nodes is represented by an encompassing circular cross-section equal to the encompassing radius times the sum of the adjacent beam element lengths projected onto a plane normal to the nodal contact force vector.

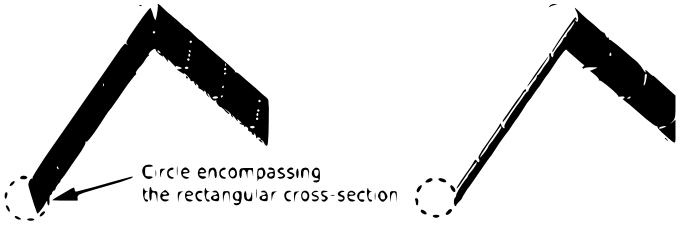


Figure 6: Nondefault circular cross-section approximation of beam elements with rectangular cross-sections.

Each nodal value of effective radius correspond to the minimum encompassing radius among adjacent beam or truss elements. Effective values of nodal radii are interpolated within a beam or truss element for the purposes of contact calculations.

Input File Usage: Use the following option to activate contact calculations based on the actual cross-section of the beam element (default):

**CONTACT CONTROLS ASSIGNMENT*, BEAM CROSS SECTION=EXACT

Use the following option to activate contact calculations based on a circumferential approximation of the beam element:

**CONTACT CONTROLS ASSIGNMENT*, BEAM CROSS SECTION=CIRCUMSCRIBED CIRCLE

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit):Contact Controls Assignment:** toggle on **Beam Cross Section assignment**, select **Circumscribed Circle** or **Exact**

Eulerian-Lagrangian Contact

The general contact algorithm also enforces contact between Eulerian materials and Lagrangian surfaces. This algorithm automatically compensates for mesh size discrepancies to prevent penetration of Eulerian material through the Lagrangian surface. The all-inclusive surface that is defined by Abaqus/Explicit can be used to enforce contact between all Eulerian materials and all Lagrangian bodies in a model; you can also specify individual Eulerian surfaces in the contact domain (see *Eulerian Surface Definition*). Eulerian-Lagrangian contact is enforced only for Lagrangian surfaces defined on solid and shell elements. Other surface types, such as beam edges and analytical rigid surfaces, are ignored. Contact interactions between Eulerian materials and interactions due to Eulerian material self-contact are handled naturally by the Eulerian formulation; these interactions do not require a general contact definition. See *Interactions* for more information.

Contact Involving DEM or SPH Particles

The general contact algorithm enforces the following types of contact involving DEM or SPH particles:

- contact between DEM or SPH particles and other Lagrangian surfaces; and
- contact among DEM particles.

See *Discrete Element Method* and *Smoothed Particle Hydrodynamics* for more information regarding contact involving DEM and SPH particles, respectively.

Including General Contact in an Analysis

If a general contact definition does not appear in a step, any general contact definition active in the previous step will be propagated to the current step.

For convenience, general contact can be defined as model data. A general contact definition specified as model data is considered to be defined in the initial step, or "Step 0," of the analysis; it can be modified or removed in Step 1 or later steps.

Input File Usage: Use the following option to indicate the beginning of a general contact definition:

**CONTACT*

This option can be used multiple times per steps, and the options will be concatenated.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**

Removing General Contact Definitions

You can remove the previously specified general contact definition and specify a new one. Contact state information (such as the proper contact normal orientation for double-sided surfaces) is transferred across step boundaries for interactions that are part of both the old and the new contact definitions. This method is the only way to reintroduce interactions that were part of prior contact exclusions.

Input File Usage: **CONTACT*, OP=NEW

Abaqus/CAE Usage: Interaction module: interaction manager: select interaction, **Deactivate**

Modifying General Contact Definitions

Alternatively, you can make changes to an existing general contact definition. In this case the existing general contact definition remains active and any additional information specified is appended to the general contact definition. Prior contact exclusions cannot be converted into contact inclusions with this method.

Contact state information (such as the proper contact normal orientation for double-sided surfaces) is transferred across step boundaries even if the contact domain is modified.

Input File Usage: **CONTACT*, OP=MOD

Abaqus/CAE Usage: Interaction module: interaction manager:
select interaction, **Edit**

Example

Each part of a general contact definition is considered independently when it is modified. For example, the following contact definition is specified in Step 1 (the individual options are discussed later in this section):

```
*CONTACT
*CONTACT INCLUSIONS
surf_1,
*CONTACT EXCLUSIONS
surf_a, surf_b
```

This contact definition is then modified in Step 2 with the following input:

```
*CONTACT, OP=MOD
*CONTACT INCLUSIONS
surf_2, surf_3
*CONTACT EXCLUSIONS
surf_a, surf_c
```

An equivalent contact definition for Step 2 could be specified as follows:

```
*CONTACT, OP=NEW
*CONTACT INCLUSIONS
surf_1,
surf_2, surf_3
*CONTACT EXCLUSIONS
surf_a, surf_b
surf_a, surf_c
```

Defining the General Contact Domain

You specify the regions of the model that can potentially come into contact with each other by defining general contact inclusions and exclusions.

All contact inclusions in an analysis are applied first, then all contact exclusions are applied, regardless of the order in which they are specified. The contact exclusions take precedence over the contact inclusions. The general contact algorithm will consider only those interactions specified by the contact inclusions definition and not specified by the contact exclusions definition.

General contact interactions typically are defined by specifying self-contact for the default automatically generated surface provided by Abaqus/Explicit. All surfaces used in the general contact algorithm can span multiple unattached bodies, so self-contact in this algorithm is not limited to contact of a single body with itself. For example, self-contact of a surface that spans two bodies implies contact between the bodies as well as contact of each body with itself.

Specifying Contact Inclusions

Define contact inclusions to specify the regions of the model that should be considered for contact purposes.

Specifying "Automatic" Contact for the Entire Model

You can specify self-contact for a default unnamed, all-inclusive surface defined automatically by Abaqus/Explicit. This default surface contains, with the exceptions noted below, all exterior element faces, all analytical rigid surfaces and all edges based on beam and truss elements in the model, as well as the nodes attached to these faces and edges; in addition, feature edges are included according to the user-specified criteria (see *Assigning Surface Properties for General Contact in Abaqus/Explicit*). This is the simplest way to define the contact domain. With this approach contact is modeled for all node-to-facet, node-to-analytical rigid surface, and edge-to-edge interactions of the nodes, facets, analytical rigid surfaces, and contact edges of the default surface. This default surface does not include the following:

- Nodes that cannot be part of an element-based surface; for example, nodes attached only to point masses or connectors.
- Faces, edges, and nodes that belong only to cohesive elements. In fact, this default surface is generated as if cohesive elements were not present. See *Modeling with Cohesive Elements* for further discussion of contact modeling issues related to cohesive elements.

Input File Usage: Use both of the following options to specify "automatic" contact for the entire model:

**CONTACT*
**CONTACT INCLUSIONS*, ALL EXTERIOR

The **CONTACT INCLUSIONS* option should have no data lines when the ALL EXTERIOR parameter is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Included surface pairs: All* with self**

Specifying Individual Contact Interactions

Alternatively, you can define the general contact domain directly by specifying the individual contact surface pairings. Self-contact will be modeled only if the two surfaces specified in a pair overlap (or are identical) and will be modeled only in the overlapping region.

Multiple surface pairings can be included in the contact domain. At least one surface in each pair must be either an element-based surface or an analytical rigid surface.

Input File Usage: Use both of the following options to specify individual contact interactions:

**CONTACT*
**CONTACT INCLUSIONS*
surface_1, *surface_2*

At least one data line must be specified when the ALL EXTERIOR parameter is omitted. Either or both of the data line entries can be left blank, but each data line must contain at least a comma; an error message will be issued for empty data lines. If the first surface name is omitted, the default unnamed, all-inclusive, automatically generated surface is assumed. If the second surface name is omitted or is the same as the first surface name, contact between the first surface and itself is assumed. Leaving both data line entries blank is equivalent to using the ALL EXTERIOR parameter.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Included surface pairs: Selected surface pairs: Edit**, select the surfaces in the columns on the left, and click the arrows in the middle to transfer them to the list of included pairs

Examples

The following input specifies that contact should be enforced between the default all-inclusive, automatically generated surface and *surface_2*, including self-contact in any overlap regions:

```
*CONTACT
*CONTACT INCLUSIONS
 , surface_2
```

Either of the following methods can be used to define self-contact for *surface_1*:

```
*CONTACT
*CONTACT INCLUSIONS
surface_1,
```

or

```
*CONTACT
*CONTACT INCLUSIONS
surface_1, surface_1
```

The following input can be used to introduce a node-based surface containing point masses to the contact domain as well as specify self-contact for the default all-inclusive, automatically generated surface:

```
*CONTACT
*CONTACT INCLUSIONS
 ,
 , node_based_surf
```

Specifying Contact Exclusions

You can refine the contact domain definition by specifying the regions of the model to exclude from contact.

The primary motivation for specifying contact exclusions is to avoid physically unreasonable contact interactions. For example, a finite element model might contain multiple forming tools, but not all the tools participate in the

forming process simultaneously; you can specify contact exclusions to prevent certain tools from participating in the contact model in certain steps.

You do not need to be concerned with specifying contact exclusions for parts of the model that are not likely to interact, since these exclusions typically will have minimal effect on computational performance.

Contact will be ignored for all the surface pairings specified, even if these interactions are specified directly or indirectly in the contact inclusions definition.

Multiple surface pairings can be excluded from the contact domain. At least one surface in each pair must be either an element-based surface or an analytical rigid surface. Keep in mind that surfaces can be defined to span multiple unattached bodies, so self-contact exclusions are not limited to exclusions of single-body contact.

You cannot exclude only one side of shell-like surfaces. If a side label (SPOS or SNEG) is used in defining an element-based shell-like surface and that surface is excluded from contact, Abaqus/Explicit will exclude all faces associated with these elements.

Input File Usage: Use both of the following options to specify contact exclusions:

**CONTACT*
**CONTACT EXCLUSIONS*
surface_1, *surface_2*

Either or both of the data line entries can be left blank. If the first surface name is omitted, the default unnamed, all-inclusive, automatically generated surface is assumed. If the second surface name is omitted or is the same as the first surface name, contact between the first surface and itself is excluded from the contact domain.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Excluded surface pairs: Edit**, select the surfaces in the columns on the left, and click the arrows in the middle to transfer them to the list of excluded pairs

Automatically Generated Contact Exclusions

Abaqus/Explicit automatically generates contact exclusions for general contact in some situations.

- Contact exclusions are generated automatically for interactions that are defined with the contact pair algorithm or surface-based tie constraints to avoid redundant (and possibly inconsistent) enforcement of these interaction constraints. For example, if a contact pair is defined for `surface_1` and `surface_2` and "automatic" general contact is defined for the entire model, Abaqus/Explicit would generate a contact exclusion for general contact between `surface_1` and `surface_2`, so that interactions between these surfaces would be modeled only with the contact pair algorithm. These automatically generated contact exclusions are in effect only during the steps in which the contact pair algorithm or surface-based tie constraint interactions are active.
- Abaqus/Explicit automatically generates contact exclusions for self-contact of each rigid body in the model, because it is not possible for a rigid body to contact itself.
- When you specify pure main-secondary contact surface weighting for a particular general contact surface pair, contact exclusions are generated automatically for the main-secondary orientation opposite to that specified (see *Contact Formulation for General Contact in Abaqus/Explicit* for more information on this type of contact exclusion).
- The general contact algorithm, unlike the contact pair algorithm, activates and deactivates contact faces and contact edges in the contact domain based on the failure status of the underlying elements. See *Modeling Surface Erosion* below for details.

Examples

The following input specifies that the contact domain is based on self-contact of an all-inclusive, automatically generated surface but that contact (including self-contact in any overlap regions) should be ignored between the all-inclusive, automatically generated surface and *surface_2*:

```
*CONTACT
*CONTACT INCLUSIONS, ALL EXTERIOR
*CONTACT EXCLUSIONS
 , surface_2
```

Either of the following methods can be used to exclude self-contact for *surface_1* from the contact domain:

```
*CONTACT EXCLUSIONS
surface_1,
```

or

```
*CONTACT EXCLUSIONS
surface_1, surface_1
```

Modeling Surface Erosion

General contact allows the use of element-based surfaces to model surface erosion for analyses that include material failure. If an appropriate “interior” or “eroding” surface is defined (as discussed in *Generating an Interior Surface Automatically*), the surface topology evolves to match the exterior of the elements that have not failed. Because of reduced memory usage, eroding surfaces are preferred over interior surfaces. Alternatively, if only one of the bodies can erode, a node-based surface can be used to model surface erosion; this approach can be used with either the general contact or contact pair algorithms. However, even if only one body can erode, it is recommended to define an element-based surface for the eroding body to avoid the usual limitations of node-based surfaces (see *Node-Based Surface Definition*).

The general contact algorithm modifies the list of contact faces and contact edges that are active in the contact domain based on the failure status of the underlying elements (element failure is discussed in *Dynamic Failure Models*). General contact considers a face only if its underlying element has not failed and it is not coincident with a face from an adjacent element that has not failed; thus, exterior faces are initially active, and interior faces are initially inactive. Once an element fails, its faces are removed from the contact domain, and any interior faces that have been exposed are activated. A contact edge is removed when all the elements that contain the edge have failed. New contact edges are not created as elements erode. Based on this algorithm, the active contact domain evolves during the analysis as elements fail (see *Figure 7* for an example of an eroding solid).

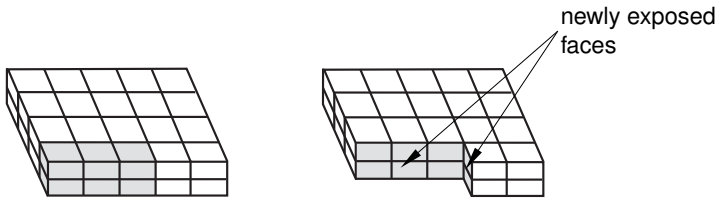


Figure 7: Topology of an eroding contact surface.

You can control whether contact nodes remain in the contact domain after all the surrounding elements have failed. By default, these nodes remain in the contact domain and act as free-floating point masses that can experience contact with faces that are still part of the contact domain. You can specify that nodes of element-based surfaces should erode (that is, be removed from the contact domain) once all contact faces and contact edges to which they are attached have eroded. Further discussion of this technique, including reasons for and against nodal erosion, can be found in *Contact Controls for General Contact in Abaqus/Explicit*.

Erosion of Surfaces Specified on Solid Elements

For a solid element mesh consisting of elements that might fail, every currently exposed face can potentially be involved in contact. Defining eroding contact surfaces and including them in the general contact domain includes the following steps:

1. Define an element set named ELERODE that contains all the solid elements in the model that refer to a material failure model.
2. Create an eroding surface named SURFERODE for this element set, as described in *Creating Surfaces on Solid, Continuum Shell, and Cohesive Elements*. The general contact algorithm activates and deactivates faces as necessary as elements fail and the free surface evolves.
3. Explicitly include this surface in the contact domain. Defining "automatic" general contact for the entire model is not sufficient because the contact domain created when this method is used does not include any interior faces. Therefore, you must define the pairwise interactions with the erodable surface explicitly in the contact inclusions definition, as outlined in *Table 1* and *Table 2*.

Table 1: Contact inclusions definitions.

Contact inclusions	Input file syntax
Self-contact for the default all-inclusive surface specifies contact between every exterior face in the model	,
Contact between the default all-inclusive surface and SURFERODE specifies contact between every exterior face and SURFERODE	, SURFERODE
Self-contact for SURFERODE specifies self-contact between the eroding bodies	SURFERODE,

Table 2: Contact inclusions definitions in Abaqus/CAE.

Contact inclusions	Abaqus/CAE syntax
Self-contact for the default all-inclusive surface specifies contact between every exterior face in the model	First Surface: (All*); Second Surface: (Self)
Contact between the default all-inclusive surface and SURFERODE specifies contact between every exterior face and SURFERODE	First Surface: (All*); Second Surface: SURFERODE
Self-contact for SURFERODE specifies self-contact between the eroding bodies	First Surface: SURFERODE; Second Surface: (Self)

Alternatively, you could create a more concise definition of the same contact domain by first defining a surface named SURFALL that includes all exterior faces in the entire model and all interior faces of element set ELERODE. In this case, since all faces (exterior and interior) in the contact domain are defined in one surface, there is no need to define contact explicitly between the exterior and interior faces. It would be adequate to specify only self-contact for SURFALL.

Abaqus/Explicit automatically computes a nonzero contact thickness associated with interior faces based on element dimensions, and this default value cannot be changed via a surface property assignment.

Erosion of Surfaces Specified on Structural Elements

For structural elements, the general contact algorithm checks the underlying elements of the faces (or "contact edges" on beam and truss elements) for failure. Once the underlying element fails, the face is removed. As with solids, feature edges on structural elements are removed once all the surrounding faces have failed. A perimeter edge (for example, on the perimeter of a shell element mesh) is removed once the face it is connected to fails. New perimeter edges are not created to conform to the new perimeter created by the removal of a face.

Output

The surfaces that compose the general contact domain are available as output in addition to the contact analysis output variables.

General Contact Domain and Component Surfaces in Abaqus/Explicit

Abaqus/Explicit generates the following internal surfaces when a general contact domain is defined:

- `General_Contact_Faces_Stepk`,
- `General_Contact_Edges_Stepk`, and
- `General_Contact_Nodes_Stepk`,

where `k` is the step number. `General_Contact_Nodes_Stepk` contains only nodes in the general contact domain that are not included in the other two surfaces. For example, `General_Contact_Faces_Step2` would contain all surface faces (interior and exterior) that were initially included in the general contact domain for Step 2. These surfaces contain the contact faces, edges, and nodes that were included in the contact domain at the beginning of the step and are not modified to reflect surface erosion.

Abaqus/Explicit generates the following internal surfaces associated with "component surfaces":

- `General_Contact_Faces_Stepk_Compm` and
- `General_Contact_Edges_Stepk_Compm`,

where *m* is the automatically assigned "component number." Each feature edge component surface, `General_Contact_Edges_Stepk_Compm`, has a subset of face edges (satisfying the feature edge criteria) of the corresponding face component surface, `General_Contact_Faces_k_Compk`. The face component surfaces have no nodes in common with each other.

Abaqus/Explicit also generates internal surfaces associated with general contact when material names are used to identify regions where nondefault contact properties or surface properties are assigned, as discussed in *Assigning Contact Properties* and *Assigning Surface Properties*. These internal surfaces are named `_MATSURF_Material Name_`, where *Material Name* corresponds to the name of the material specified for the property assignment.

Internal surfaces can be viewed using display groups in the Visualization module of Abaqus/CAE (see the *Introduction*). The internal surface names used by Abaqus/Explicit should not appear in the input file.

General Contact Output Variables

You can write the contact surface variables associated with general contact interactions to the Abaqus output database (`.odb`) file (see *Writing Surface Output to the Output Database* for more information). The available variables are contact pressure, normal contact force, frictional force, and whole surface resultant quantities (that is, force, moment, center of pressure, and total area in contact).

Field Output

The generic variables CSTRESS and CFORCE are valid field output requests for general contact in Abaqus/Explicit. If CSTRESS is requested for the general contact domain, the variables CPRESS (contact pressure) and CSHEARMAG (magnitude of the frictional shear stress) are available in the output database and can be contoured in Abaqus/CAE. If CFORCE is requested for the general contact domain, the variables CNORMF (normal contact force) and CSHEARF (shear contact force) are available in the output database and can be plotted as vectors in a symbol plot in Abaqus/CAE.

For general contact CPRESS is calculated as the magnitude of the net contact normal force (the CNORMF vector) per unit area (it is an unsigned value). This convention for reporting contact pressure is different from

the convention used for contact pairs. The direction of action of the net contact pressure for general contact can be determined by examining a plot of CNORMF.

CNORMF and CSHEARF are resultant force quantities. If a double-sided surface is contacted on both sides, the resultant force is a vector sum of the force from each side of the surface (for example, the contact normal force will be zero for a double-sided surface that is pinched with equal and opposite forces on each side of the surface).

Some small noise in contact stress and contact force output is likely near the perimeters of thin shells. This behavior is expected and does not negatively impact the overall solution.

Contact stresses for surfaces based on the second-order tetrahedral element type C3D10 are smoothed during postprocessing operations. Therefore, inferring contact force from contact stresses can be somewhat misleading for this element type; instead, you can request nodal consistent contact force output (CFORCE), which represents the contact force distribution.

Displacement field output (U) for the entire model is written to the output database automatically when any of the contact field output variables are requested.

Several output variables associated with quantities computed at secondary nodes or edge nodes are also available, with generic output variable names CDISP, CDISPR, CSLIPR, and CTANDIR. These output variables are not available for Eulerian-Lagrangian contact or contact involving particles. If these generic output variables names are requested, the specific output variables written as field output are as follows:

- Contact “displacements” (opening distance and accumulated slips) CDISP: COPEN, CSLIPEQ, CSLIP1, and CSLIP2;
- Contact opening rate and slip rates CDISPR: COPENR, CSLIPRMAG, CSLIPR1, and CSLIPR2;
- Contact slip rates CSLIPR: CSLIPRMAG, CSLIPR1, and CSLIPR2;
- Contact tangent directions CTANDIR: CTANDIR1, and CTANDIR2.

COPEN is reported only for secondary or edge nodes in contact or very close to being in contact. The accumulated slip variables remain constant when a node is out of contact. The slip rate and tangent direction output variables are reported only for secondary or edge nodes in contact. CSLIPEQ represents the total slip length at a secondary or edge node while in contact. Incremental contributions to CSLIP1 and CSLIP2 are computed as the scalar product of the incremental relative nodal displacement vector and the respective local tangent direction, $\mathbf{t}_1$ (CTANDIR1) or $\mathbf{t}_2$ (CTANDIR2).

The algorithm used to establish and evolve local tangent directions for general contact is described in *Local Tangent Directions for Contact*. As local tangent directions for contact evolve across increments, previously accumulated slip components are resolved into the new local system before incremental contributions are added to them.

For two-dimensional or axisymmetric analyses, the components in the local tangent 2-direction for the contact output variables such as CSLIP2, CSLIPR2, and CTANDIR2 have no physical meaning and, therefore, are not provided.

History Output

Several whole surface contact force-derived variables are available as history output. You can specify the surface from which the contact force resultants will be calculated.

Force distributions on the surface due to general contact are used to calculate the surface force resultants; forces due to contact pair interactions are not included and must be output separately. The contact state of a surface is output as a set of force (CFN, CFS, and CFT) and moment (CMN, CMS, and CMT) resultants with respect to the origin. Additional variables give the center of force (XN, XS, and XT) on the surface (defined as the point closest to the centroid of the surface that lies on the line of action of the resultant force for which the resultant moment is minimal). The last letter of each variable name denotes which contact force distribution on the surface

is used to calculate the resultant: the letter N denotes that the normal contact forces are used to derive the resultant quantity; the letter S denotes that the shear contact forces are used to derive the resultant quantity; and the letter T denotes that the sum of the normal and shear contact forces are used to derive the resultant quantity.

Each total moment output variable will not necessarily equal the cross product of the respective center of force vector and resultant force vector. Forces acting on two different nodes of a surface might have components acting in opposite directions, such that these nodal force components generate a net moment but not a net force; therefore, the total moment might not arise entirely from the resultant force. The center of force output variables tend to be most meaningful when the surface nodal forces act in approximately the same direction.

The total area in contact at a given time can be requested using output variable CAREA, defined as the sum of all the facets where there is contact force. The contact area reported by CAREA is generally slightly larger than the true contact area for reasonably meshed contact surfaces; therefore, interpretation of CAREA should be done with care. The discrepancy between the CAREA output and the true contact area decreases as the mesh density increases. Using contact inclusions or exclusions to limit CAREA output to smaller contact surfaces might also reduce the discrepancy in some cases. Since the CAREA output is an approximation of the true contact area, deriving force or stress values using this output might not yield accurate values; requesting contact force and stress directly is the most appropriate way to obtain accurate results.

Requesting Element Output When Modeling Surface Erosion

When modeling the erosion of surfaces, it is useful to request additional element field output of the element status (output variable STATUS). Failed elements (with an element status of zero) can then be excluded from the display group in the Visualization module of Abaqus/CAE so that the active contact surface can be identified and contact results on the active contact surface can be viewed.

Output for Beams with Actual Beam Cross-Section Representation

When you enable actual beam cross-section representation for beams with noncircular cross-sections (see *Contact Surface Representation for Beams and Trusses*), Abaqus uses an automatically generated mesh of quadrilateral surface elements to represent the contact surface. Output variables associated with contact are available on the quadrilateral surface elements for postprocessing; however, they are not available on the corresponding beam elements. Output variables such as stress and strain at integration points are available on the beam elements rather than the surface elements. For an example, see *Visualizing beam outputs with actual beam cross-section representation for contact*.

Extending the Range for Which Contact Opening Output Is Provided for Gaps

To reduce computational costs, detailed computations to monitor potential points of interaction are avoided by default where surfaces are separated by a distance greater than the minimum gap distance at which contact forces (or thermal fluxes, for example) might be transmitted. Therefore, contact opening (COPEN) output is typically not provided where surfaces are opened by more than a small amount compared to surface facet dimensions. You can extend the range for which Abaqus provides contact opening output; COPEN will be provided up to gap distances equal to a specified “tracking thickness.” Using this control might increase computational cost due to extra contact tracking computations, especially if you specify a large tracking thickness value. The tracking range will be at least as large as the specified tracking thickness value; an internally computed default will remain in effect if this value is less than the default.

You can control the tracking thickness by defining the tracking thickness surface pairwise through contact controls, which is the recommended method. Alternatively, you can define the tracking thickness through a surface interaction property.

Input File Usage: To define the tracking thickness surface pairwise through contact controls (recommended method), use the following options:

**CONTACT*
**CONTACT CONTROLS ASSIGNMENT*, TYPE=TRACKING THICKNESS
surface_1, surface_2, *value*

To define the tracking thickness through a surface interaction property, use the following option:

**SURFACE INTERACTION*, TRACKING THICKNESS=*value*

Abaqus/CAE Usage: Defining the tracking thickness through contact controls is not supported in Abaqus/CAE.

To define the tracking thickness through a surface interaction property, use the following option:

Interaction module: contact property editor: **Mechanical->Geometric Properties**: toggle on **Thickness to determine the contacting surfaces to be tracked:** *value*

Assigning Surface Properties for General Contact in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *About General Contact in Abaqus/Explicit*
- **CONTACT*
- **SURFACE PROPERTY ASSIGNMENT*
- *Specifying surface property assignments for general contact*

Overview

Surface property assignments:

- can be used to change the contact thickness used for regions of a surface based on structural elements or to add a contact thickness for regions of a surface based on solid elements;
- can be used to specify surface offsets for regions of a surface based on shell, membrane, rigid, and surface elements;
- can be used to specify which edges of a model should be included in the general contact domain;
- can be used to specify geometric corrections for regions of a surface;
- can be used to assign a coordinate system for local tangent directions to the surface and/or specify preferential frictional directions to the surface in the context of anisotropic friction;
- can be used to assign surface-based friction coefficients, such that friction coefficients for interactions can be approximated from surface-based friction coefficients; and
- can be applied selectively to particular regions within a general contact domain.

Assigning Surface Properties

You can assign nondefault surface properties to surfaces involved in general contact interactions. These properties are considered only when the surfaces are involved in general contact interactions; they are not considered when the surfaces are involved in other interactions such as contact pairs. The general contact algorithm does not consider surface properties specified as part of the surface definition. The regions with nondefault surface properties are identified with surface names or material names. For example, surface property `SurfProp_A` can assign a nondefault surface thickness to surface `Surf_1` or to the surface whose underlying elements have a section assignment with material `Rubber`. Material names cannot be used to assign geometric corrections.

Surface property assignments propagate through all analysis steps in which the general contact interaction is active.

The surface names used to specify the regions with nondefault surface properties do not have to correspond to the surface names used to specify the general contact domain. In many cases the contact interaction will be defined for a large domain, while nondefault surface properties will be assigned to a subset of this domain. Any surface property assignments for regions that fall outside the general contact domain will be ignored. The last assignment will take precedence if the specified regions overlap.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY

This option must be used in conjunction with the **CONTACT* option. It should appear at most once per step for each value of the PROPERTY parameter discussed below;

the data line can be repeated as often as necessary to assign surface properties to different regions.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**

Surface Thickness

The default calculation of the nodal surface thickness (described in detail below), which remains constant during the simulation, is appropriate for most analyses. For cases in which changes in the surface thickness might significantly affect contact behavior, Abaqus/Explicit provides options to consider the current thickness (for example, to account for effects of thinning or thickening of shell thickness) or to consider only thinning effects. You can specify a value for the surface thickness. For example, you can assign a nonzero thickness to solid element surfaces to model the effect of a finite-thickness surface coating. *Element-Based Surface Definition* contains information on the spatial variation of the surface thickness.

Specifying that original, current, or monotonically decreasing thickness should be used results in a zero thickness for node-based surfaces. You can specify a nonzero thickness for a node-based surface used with the general contact algorithm (the contact pair algorithm will not consider a nonzero thickness for such surfaces).

The general contact algorithm requires that the contact thickness does not exceed a certain fraction of the surface facet edge lengths or diagonal lengths. This fraction generally varies from 20% to 60% based on the geometry of the element. The general contact algorithm scales back the contact thickness automatically where necessary without affecting the thickness used in the element computations for the underlying elements. Abaqus/Explicit provides diagnostic information in the status (`.sta`) file if it performs such scaling.

To bypass this limitation on thickness, you can model the contact surface with surface elements (see *Surface Elements*). You must attach the surface elements to the underlying elements using a surface-based tie constraint (see *Mesh Tie Constraints*), and you must associate a physically reasonable mass with the surface elements. This requires a significant fraction of the mass to be transferred to the surface elements from the underlying elements without appreciably altering the bulk mass properties. Alternatively, you can use contact controls settings to limit the thickness reduction checks (see *Contact Controls for General Contact in Abaqus/Explicit*).

The “bull-nose” effect that occurs at shell perimeters with the contact pair algorithm (see *Assigning Surface Properties for Contact Pairs in Abaqus/Explicit*) is avoided with the general contact algorithm by default. Shell element edges, nodes, and facets reflect the shell thickness in the normal direction only and do not extend past the perimeter. You can use contact controls settings to turn off the bull-nose prevention checks (see *Contact Controls for General Contact in Abaqus/Explicit*).

Using the Original Parent Element Thickness

By default, the nodal thickness for surfaces based on shell, membrane, or rigid elements equals the minimum original thickness of the surrounding elements (see *Figure 1* and *Table 1*). If a node is shared by shell and beam elements, the contact thickness that takes precedence is the one derived from the shell element. To account for thick beams that are colocated with shell edges, the beam elements must be attached to the shell edges using a tie constraint (see *Mesh Tie Constraints*).

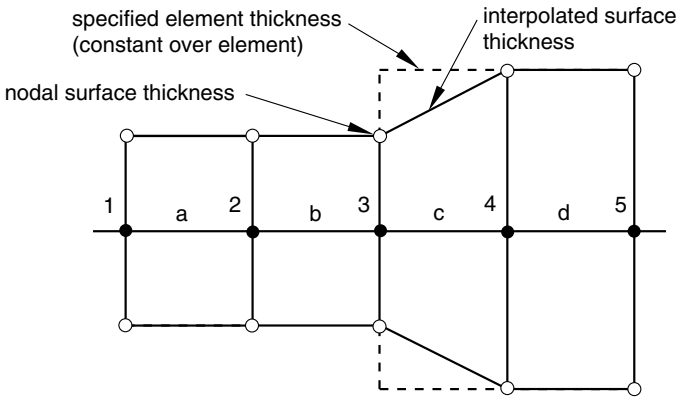


Figure 1: Continuous variation of surface thickness across facet boundaries.

Table 1: Thicknesses corresponding to figure showing continuous variation.

Node	Element	Specified element thickness	Nodal surface thickness (minimum of adjacent element thicknesses)
1			0.5
	a	0.5	
2			0.5
	b	0.5	
3			0.5
	c	0.9	
4			0.9
	d	0.9	
5			0.9

The surface thickness within a facet is interpolated from the nodal values; the interpolated surface thickness never extends past the specified element or nodal thickness, which might be significant with respect to initial overclosures. The default nodal surface thickness is zero for regions of a surface based on solid elements. If a spatially varying nodal thickness is defined for the underlying elements (see *Nodal Thicknesses*), the nodal surface thickness might not correspond exactly to the specified nodal thickness (see node 4 in *Figure 2* and *Table 2*).

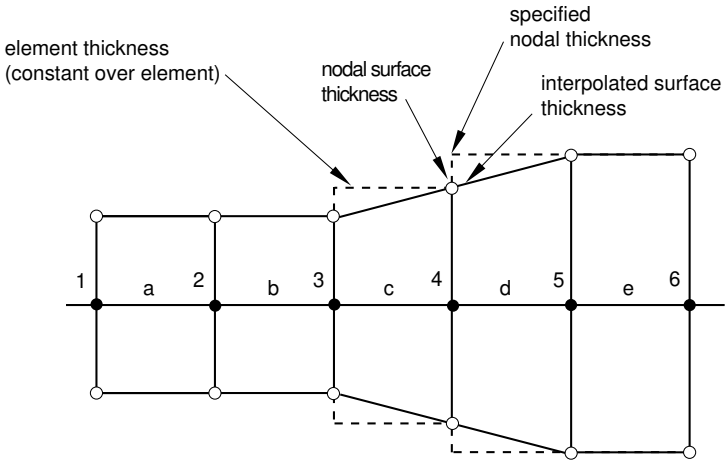


Figure 2: Small discrepancy between the nodal surface thickness and the specified nodal thickness.

Table 2: Thicknesses corresponding to figure showing small discrepancies.

Node	Element	Specified nodal thickness	Element thickness (average of specified nodal thickness)	Nodal surface thickness (minimum of adjacent element thicknesses)
1		0.5		0.5
	a		0.5	
2		0.5		0.5
	b		0.5	
3		0.5		0.5
	c		0.7	
4		0.9		0.7
	d		0.9	
5		0.9		0.9
	e		0.9	
6		0.9		0.9

The nodal surface thickness distribution will tend to be more diffuse than the specified nodal thickness distribution (because the specified nodal thicknesses are averaged to compute the element thicknesses, and the minimum of the surrounding element thicknesses is the nodal surface thickness).

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=THICKNESS
surface or *material*, ORIGINAL (default), , SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The fourth entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Surface thickness assignments: Edit**:

Select surface or material, click the arrows to transfer surface or material to list of thickness
assignments, and enter ORIGINAL in the **Thickness** column.

Using the Current Parent Element Thickness

If you specify that the current parent element thickness should be used, increases and decreases in the parent element thickness are reflected in the contact surface thickness. An upper bound limiting value for the contact surface thickness remains in effect (based on a certain fraction of the surface facet edge lengths or diagonal lengths), and the original surface thickness acts as an upper bound for perimeter nodes except when acting as main nodes of node-to-surface contact constraints. No lower bound limiting value to the contact surface thickness exists.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=THICKNESS
surface or *material*, CURRENT, , SURFACE (default) or MATERIAL

If you omit the first entry, Abaqus/Explicit assumes a default surface that encompasses the entire general contact domain.

The fourth entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus/Explicit assumes that a surface name is used.

Abaqus/CAE Usage: Defining the current parent element thickness is not supported in Abaqus/CAE.

Using the Decreasing Parent Element Thickness

If you specify that the decreasing parent element thickness should be used, only decreases in the parent element thickness are reflected in the contact surface thickness; if the parent element thickness actually increases during the analysis, the contact thickness will remain constant.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=THICKNESS
surface or *material*, THINNING, , SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The fourth entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Surface thickness assignments: Edit**:
Select surface or material, click the arrows to transfer surface or material to list of thickness
assignments, and enter THINNING in the **Thickness** column.

Specifying a Value for the Surface Thickness

You can directly specify the surface thickness value.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=THICKNESS
surface or *material*, *value*, , SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The fourth entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Surface thickness assignments: Edit**:
Select surface or material, click the arrows to transfer surface or material to list of thickness
assignments, and enter a value for the surface thickness magnitude in the **Thickness** column.

Applying a Scale Factor to the Surface Thickness

You can apply a scale factor to any value of the surface thickness. For example, if you specify that the decreasing parent element thickness should be used for `surf1` and apply a scale factor of 0.5, a value of one half the decreasing parent element thickness will be used for `surf1` when it is involved in a general contact interaction (all other surfaces included in the general contact domain will use the default original parent element thickness). Scaling the surface thickness in this way can be used to avoid initial overclosures in some situations. Abaqus/Explicit will automatically adjust surface positions to resolve initial overclosures (see *Contact Initialization for General Contact in Abaqus/Explicit*). However, if nodal position adjustments are undesirable (for example, if they would introduce an imperfection in an otherwise flat part, resulting in an unrealistic buckling mode), you might prefer to reduce the surface thickness and avoid the overclosures entirely.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=THICKNESS
surface or *material*, *value or label*, *scale_factor*, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The fourth entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Surface thickness assignments: Edit**:
Select surface or material, click the arrows to transfer surface or material to list of thickness
assignments, and enter a **Scale Factor**.

Surface Offset

A surface offset is the distance between the midplane of a thin body and its reference plane (defined by the nodal coordinates and element connectivities). It is computed by multiplying the offset fraction (specified as a fraction of the surface thickness) by the surface thickness and the element facet normal. This defines the position of the midsurface and, thus, the position of the body with respect to the reference surface; the coordinates of the nodes on the reference surface are not modified. Surface offsets can be specified only for surfaces defined on shell and similar elements (that is, membrane, rigid, and surface elements). Surface offsets specified for other elements (for example, solid or beam elements) will be ignored. By default, surface offsets specified in element section definitions will be used in the general contact algorithm.

The surface offset at each node is the average of the maximum and minimum offsets among the faces connected to the node. The offset at a point within a facet is interpolated from the nodal values. *Figure 3* shows some examples of the positioning of the contact surface with respect to the reference surface for various combinations of surface offsets. Surface offsets used in the general contact algorithm are constrained to lie between −0.5 and 0.5 of the thickness.

You specify the surface offset as a fraction of the surface thickness. The surface offset fraction can be set equal to the offset fraction used for the surface's parent elements or to a specified value. Surface offsets specified for general contact do not change the element integration.

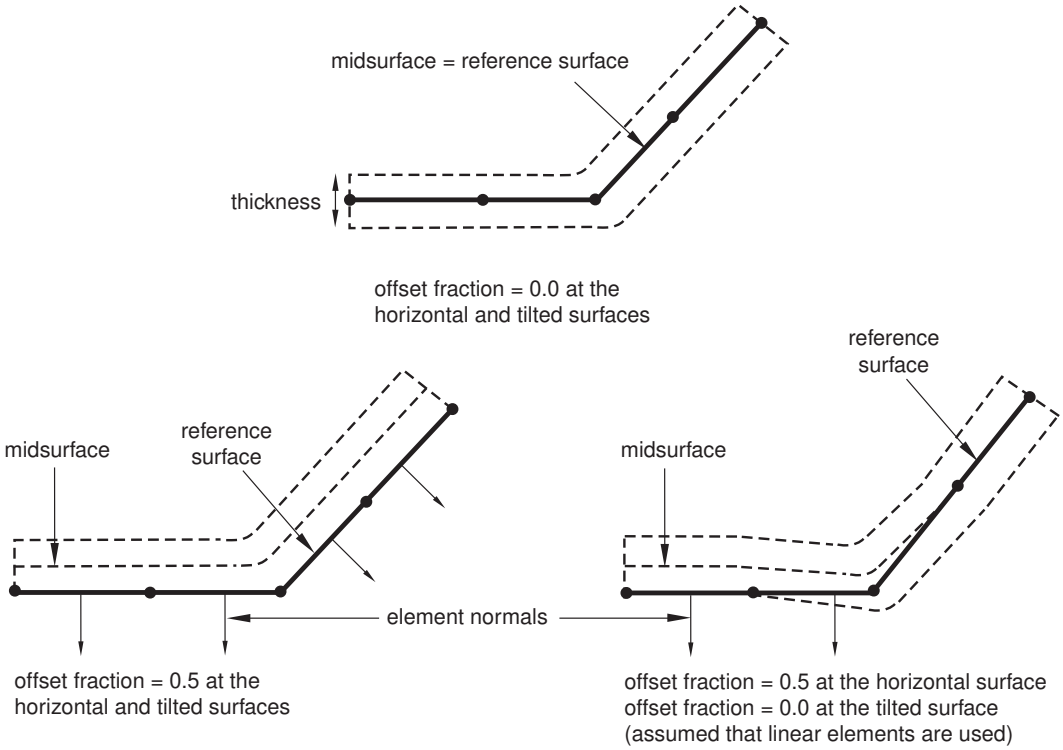


Figure 3: Specifying surface offsets for general contact.

Input File Usage: Use the following option to use the surface offset fraction from the surface's parent elements (default):

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=OFFSET FRACTION
surface or *material*, ORIGINAL, SURFACE (default) or MATERIAL

Use the following option to specify a value for the surface offset fraction:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=OFFSET FRACTION
surface or *material*, *offset*, SURFACE (default) or MATERIAL

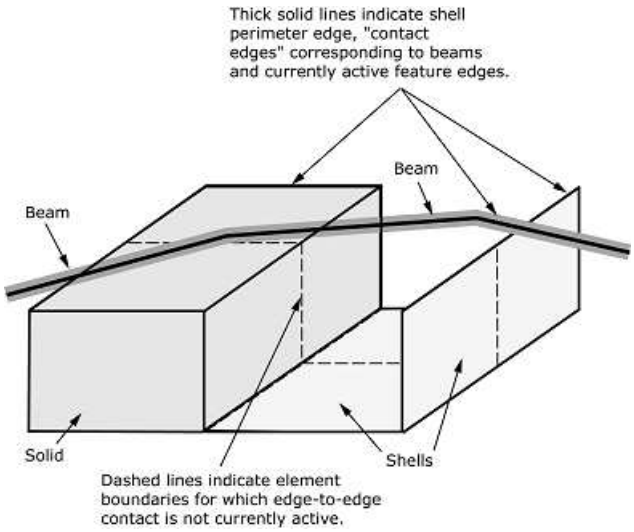
The offset can be specified as a value or a label (SPOS or SNEG). Specifying SPOS is equivalent to specifying a value of 0.5; specifying SNEG is equivalent to specifying a value of −0.5.

The third entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Shell/Membrane offset assignments: Edit**:
Select surface or material, and click the arrows to transfer surface or material to list of offset
assignments.
In the **Offset Fraction** column, enter ORIGINAL to use the surface offset fraction from the surface's parent elements, enter SPOS to use a surface offset fraction of 0.5, enter SNEG to use a surface offset fraction of −0.5, or enter a value for the surface offset fraction.

Feature Edges

Feature edges of a model are defined on beam and truss elements and edges of faces (perimeter and otherwise) of solid and structural elements. Feature edges, such as shown in *Figure 4*, can participate in edge-to-edge contact in Abaqus/Explicit (see *Surfaces Used for General Contact*).



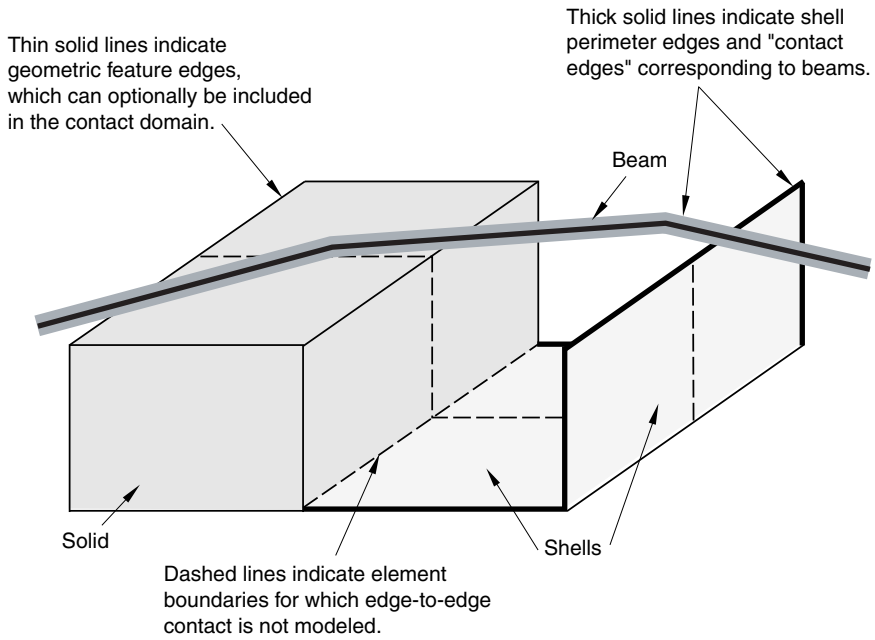


Figure 4: General contact domain, including edge-to-edge contact.

By default in Abaqus/Explicit:

- “Contact edges” of beam and truss elements and perimeter edges of shells and membranes act as primary feature edges (see *Primary and Secondary Feature Edges*), as long as the underlying elements remain active.
- Feature angle thresholds of 40° for primary feature edges and 30° for secondary feature edges are applied dynamically throughout a simulation to determine which edges of solid elements and which non-perimeter edges of shell elements currently act as primary or secondary feature edges. The feature angle is the angle formed between the normal directions of the two facets connected to an edge, as discussed further in *The Feature Angle*. As an edge’s feature angle evolves during a simulation, its classification as a primary feature edge, a secondary feature edge, or not a feature edge might also change. *Figure 5* shows a crumpling example in which many feature edges form during a simulation. Other types of simulations (such as airbag deployment) involve many feature edges unfolding over the course of a simulation.

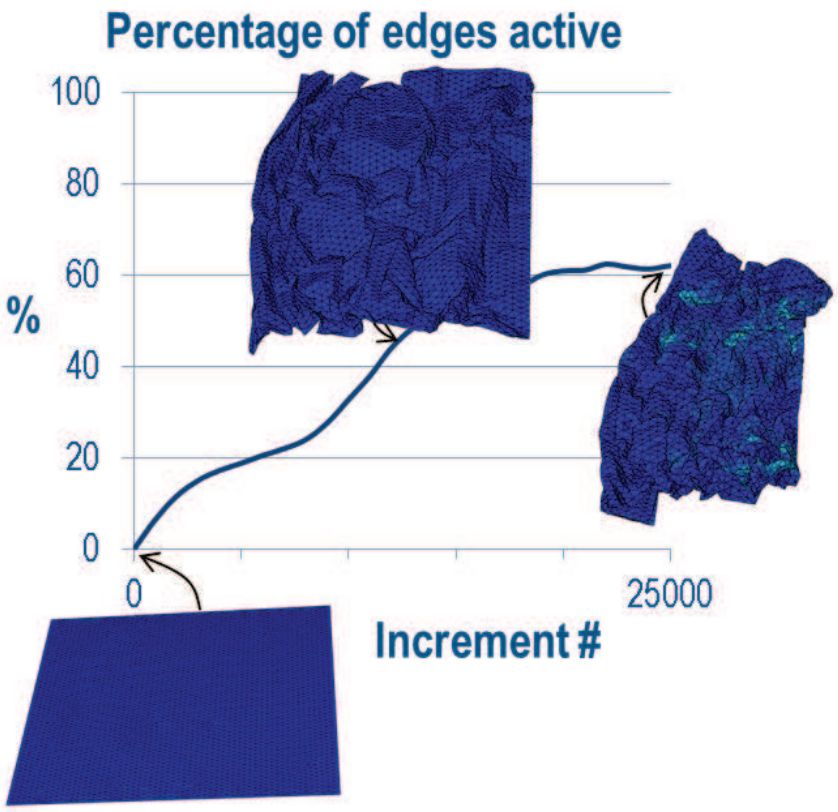


Figure 5: Percentage of active edges versus increment number for a paper crumpling example: original (flat sheet), intermediate, and final configurations.

Input File Usage: By default, Abaqus/Explicit automatically applies evolving feature edge criteria to the entire contact domain, including all structural and solid elements. This is equivalent to the following definition:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA
surface or *material*,,,, CURRENT,, SURFACE (default) or MATERIAL

The seventh entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**: select surface or material and the configuration CURRENT in the columns on the left, and click the arrows to transfer the surface or material to the list of feature assignments.

Using a Fixed Set of Active Feature Edges for Contact Based on Original Feature Angles

Optionally, Abaqus/Explicit can establish a fixed set of active feature edges for contact based on original feature angles. If no feature angle thresholds are specified explicitly, the list of active feature edges matches the default initially active feature edges for dynamically applied criteria (40° for primary feature edges and 30° for secondary feature edges), but this list is not updated during the simulation. This option is not well suited for common scenarios involving significant deformation during a simulation. However, using a fixed set of active feature edges can save computational time for simulations involving small deformation.

Input File Usage: Use the following option to apply the feature edge criteria only at the beginning of the simulation such that the list of active feature edges for contact remains constant:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA
surface or *material*,,,, ORIGINAL,, SURFACE (default) or MATERIAL

The seventh entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select the surface or material and the configuration ORIGINAL in the columns on the left, and click the arrows to transfer the surface or material to the list of feature assignments.

Limiting Feature Edges to Perimeter Edges and Contact Edges of Beams and Trusses

You can limit feature edges for edge-to-edge contact to perimeter edges and contact edges of beams and trusses. Perimeter edges occur on "physical" perimeters of shell elements and on "artificial" edges that occur when a subset of exposed facets on a body are included in the general contact domain. When structural elements share nodes with continuum elements, the perimeter edges are not activated on the structural elements because the criterion to designate them as such is no longer satisfied.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA
surface or *material*, PERIMETER EDGES (default),,,,, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The seventh entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select surface or material, click the arrows to transfer surface or material to list of feature
assignments, and enter PERIMETER in the **Primary Feature Edge Criteria** column.

Specifying Particular Feature Edges to Be Activated

You can choose particular feature edges on surface, structural, and rigid elements to be activated in domain. A surface containing a list of element labels and edge identifiers (see "Defining edge-based surfaces" in *Element-Based Surface Definition*) is used to specify the edges to activate.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA
surface or *material*, PICKED EDGES,,,,, SURFACE (default) or MATERIAL

The seventh entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select the surface or material, click the arrows to transfer the surface or material to the list of feature
assignments, and enter PICKED in the **Primary Feature Edge Criteria** column.

Specifying That All Feature Edges Should Be Activated

You can choose to activate all edges each increment in a given surface in the general contact domain. However, this option degrades performance.

Input File Usage: Use the following option to force all edges to be considered each increment:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA
surface or *material*, ALL EDGES,,, ORIGINAL,, SURFACE (default) or MATERIAL

The seventh entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select the surface or material, click the arrows to transfer the surface or material to the list of feature
assignments, and enter ALL in the **Primary Feature Edge Criteria** column.

Specifying That All Feature Edges Should Be Deactivated

You can choose to deactivate all feature edges (including perimeter edges) in the general contact domain. This option does not deactivate “contact edges” associated with beam and truss elements.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA
surface or *material*, NO FEATURE EDGES,,,,, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The seventh entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select the surface or material, click the arrows to transfer the surface or material to the list of feature
assignments, and enter NONE in the **Primary Feature Edge Criteria** column.

Specifying a Cutoff Feature Angle

If you specify a cutoff feature angle as the feature edge criteria, perimeter edges and geometric edges with feature angles greater than or equal to the specified angle are activated in the general contact domain. By default, the feature angle thresholds are applied dynamically throughout the simulation. Optionally, you can specify that the feature angle thresholds are applied only once at the beginning of the analysis. As described previously, you can activate additional feature edges if required.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA
surface or *material*, *feature_angle_value*,,,,, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The fifth entry indicates how to apply the feature angle thresholds: CURRENT to apply them throughout the simulation (default) or ORIGINAL to apply them only once at the beginning of the analysis.

The seventh entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select surface or material, click the arrows to transfer surface or material to list of feature
assignments, and enter a value for the cutoff feature angle (in degrees) in the **Primary Feature Edge Criteria** column.

Example: Assigning Different Feature Edge Criteria to Different Regions

You can assign a different feature edge criteria to different regions of the general contact domain. For example, *Table 3*and *Table 4* show the input that could be used to specify that none of the feature edges of `surf1`, only perimeter edges of `surf2`, and perimeter edges and feature edges of `surf3` with a feature angle greater than 30° should be considered for edge-to-edge contact.

Table 3: Feature edge criteria example: input file.

Input File Syntax
`surf1, NO FEATURE EDGES`
`surf2, PERIMETER EDGES`
`surf3, 30`

Table 4: Feature edge criteria example: Abaqus/CAE.

Abaqus/CAE Syntax
Surface: `surf1`, Feature Edge Criteria: NONE
Surface: `surf2`, Feature Edge Criteria: PERIMETER
Surface: `surf3`, Feature Edge Criteria: 30

Primary and Secondary Feature Edges

To reduce computational cost in certain situations, it might be desirable to specify two feature angle criteria for a given surface. Edges satisfying the more restrictive criteria are considered primary feature edges, and edges satisfying the less restrictive criteria only are considered secondary feature edges. If primary and secondary feature edge criteria are in effect, Abaqus/Explicit enforces edge-to-edge contact between primary feature edges and between primary feature edges and secondary feature edges only. Edge-to-edge contact is not enforced between secondary feature edges. This ensures that interpenetrations are avoided at locations where there are "true" edges in the model, without the need to activate primary feature edges at locations where the gradients in

the surface normals are only moderate. A judicious choice of criteria for selecting primary and secondary feature edges can lead to significant savings in computational costs.

Secondary feature edges can be selected for a surface by specifying a secondary feature edge criterion in addition to the criterion used to select the primary feature edges for that surface. If the secondary feature edge criterion is omitted, only primary feature edges are activated for the surface. Allowable criteria for secondary feature edges are:

- all edges that have not been selected as primary feature edges;
- all picked edges that have not been selected as primary feature edges;
- all perimeter edges that have not been selected as primary feature edges; and
- all edges with a feature angle greater than a specified cutoff angle value that have not been selected as primary feature edges.

The allowable values for the secondary feature edge criterion permit possible combinations of criteria for primary feature edges and secondary feature edges, shown in *Table 5*.

Table 5: Valid combinations of primary feature edge and secondary feature edge criteria.

Primary Feature Edge Criterion	Secondary Feature Edge Criterion
No feature edges	All remaining edges, picked edges, perimeter edges, cutoff angle
All edges	Any criterion specified for secondary feature edges will be ignored
Picked edges	All remaining edges, perimeter edges, cutoff angle
Perimeter edges	All remaining edges, picked edges, cutoff angle
Cutoff angle	All remaining edges, picked edges, perimeter edges, cutoff angle

Specifying All Remaining Edges as Secondary Feature Edges

You can specify that all edges belonging to the surface that have not been selected as primary feature edges become secondary feature edges.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA
surface or *material*, *primary feature edge criterion*, ALL REMAINING EDGES,,,,
SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The seventh entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select the surface or material, click the arrows to transfer the surface or material to the list of feature
assignments, and enter ALL in the **Secondary Feature Edge Criteria** column.

Specifying Picked Edges as Secondary Feature Edges

You can specify that all picked edges of the surface that have not already been selected as primary feature edges become secondary feature edges.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA *surface* or *material*, *primary feature edge criterion*, PICKED EDGES,,,, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The seventh entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select the surface or material, click the arrows to transfer the surface or material to the list of feature
assignments, and enter PICKED in the **Secondary Feature Edge Criteria** column.

Specifying Perimeter Edges as Secondary Feature Edges

You can specify that all perimeter edges of the surface that have not already been selected as primary feature edges become secondary feature edges.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA *surface* or *material*, *primary feature edge criterion*, PERIMETER EDGES,,,, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The seventh entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select surface or material, click the arrows to transfer surface or material to list of feature
assignments, and enter PERIMETER in the **Secondary Feature Edge Criteria** column.

Specifying a Cutoff Feature Angle for Secondary Feature Edges

You can specify that edges on the surface with a feature angle greater than the specified value that have not been selected as primary feature edges become secondary feature edges. If an angle value has also been specified for primary feature edges, the angle value specified for secondary feature edges must be smaller than the value specified for primary edges.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA *surface* or *material*, *primary feature edge criterion*, *feature_angle_value*,,,, SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The seventh entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select surface or material, click the arrows to transfer surface or material to list of feature
assignments, and enter a value for the cutoff feature angle (in degrees) in the **Secondary Feature Edge Criteria** column.

Specifying That Edges Are Activated Only as Secondary Feature Edges

For a particular surface you might not want to activate any primary feature edges; instead, you might want to activate all or some edges on the surface as secondary feature edges (to enforce contact between these secondary feature edges and primary feature edges on another surface in the model). In that case you can specify that no feature edges should be activated as the primary feature edge criterion for the surface, while using any criterion of choice for the secondary feature edges.

Input File Usage: **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA
surface or *material*, NO FEATURE EDGES, *secondary feature edge criterion*,,,,
SURFACE (default) or MATERIAL

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

The seventh entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**:
Select the surface or material, click the arrows to transfer the surface or material to the list of feature
assignments, and enter NONE in the **Secondary Feature Edge Criteria** column.

The Feature Angle

The feature angle is the angle formed between the normals of the two facets connected to an edge. By default, the angles between facets are based on the initial configuration. However, the most efficient approach for accurately resolving contact is often to apply the feature edge criteria to the current configuration. In this case the edges that are eligible for edge-to-edge contact evolve during the simulation.

A negative angle will result at concave meetings of facets; therefore, these edges are not included in the contact domain if the feature edge criteria is based on a cutoff feature angle. *Figure 6* shows some examples of how the feature angle is calculated for different edges.

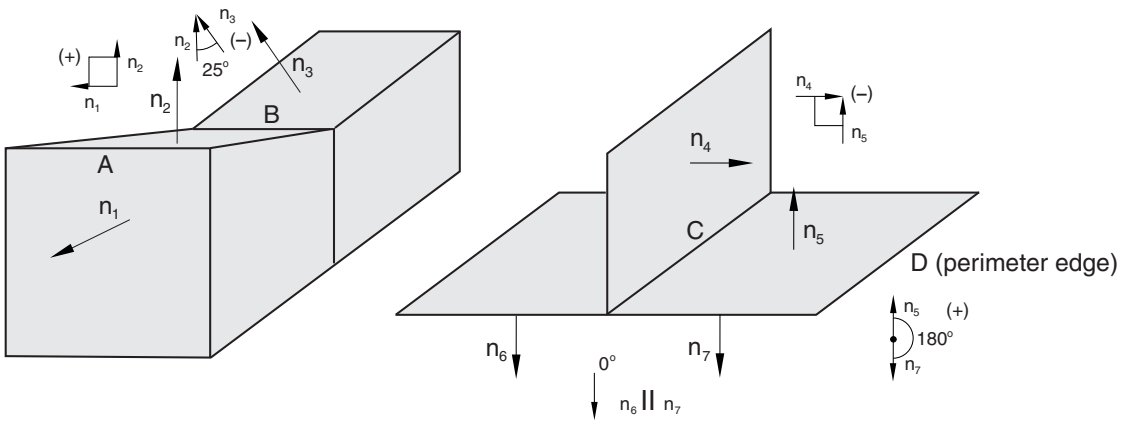


Figure 6: Calculating the feature angle.

The feature angle for edge A is 90° (the angle between $\boldsymbol{n_1}$ and $\boldsymbol{n_2}$); the feature angle for edge B is −25° (the angle between $\boldsymbol{n_2}$ and $\boldsymbol{n_3}$). Edge C forms a T-intersection with three facets (shown in two dimensions in *Figure 7*); its feature angles are 0°, −90°, and −90°.

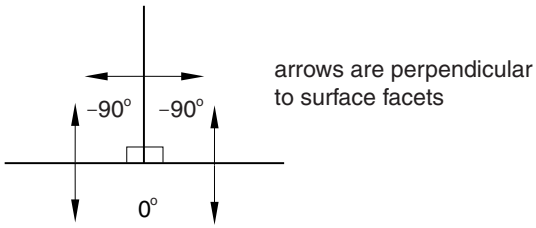
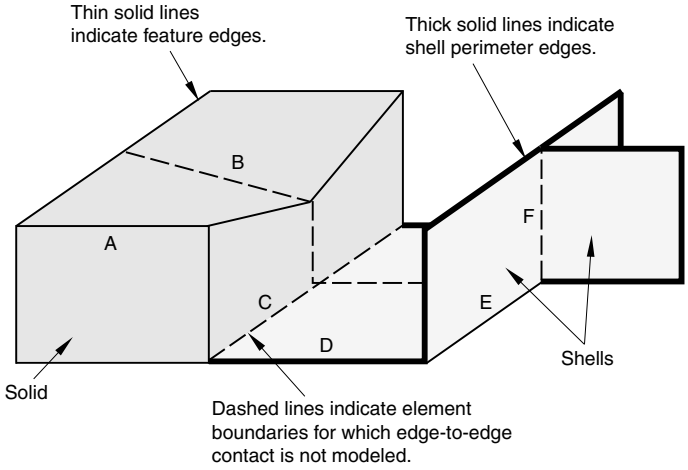


Figure 7: Feature angles for a T-intersection.

Perimeter edges (for example, edge D in *Figure 6*) can be thought of as a special type of feature edge where the feature angle is 180°.

The sign of the feature angle is considered when determining whether or not a geometric feature edge should be activated in the general contact domain. For example, if a cutoff feature angle of 20° were specified, edge A would be activated as a feature edge in the contact model (90° > 20°) but edges B and C would not be activated: −25° < 20° and 0° (the maximum feature angle for edge C) < 20°.

Figure 8 illustrates further how the feature angle is used to determine which geometric feature edges should be activated in the general contact domain.



Edge	Largest feature angle at edge	Other feature angles at edge
A	approximately $+105^{o}$	none
B	approximately -30^{o}	none
C	0^{o}	-90^{o}
D	$+180^{o}$	none
E	$+90^{o}$	-90^{o}
F	0^{o}	$-90^{o}, -90^{o}$

Figure 8: Feature edges activated in the general contact domain for a cutoff feature angle of 20°.

The table to the right of the figure lists the feature angle values for various edges in the model. Edges connected to more than two facets, as well as edges connected to two shell facets, have more than one corresponding feature angle. The largest feature angle at an edge is compared to the specified cutoff feature angle. For example, if a cutoff feature angle of 20° were specified, edges A, D, and E would be considered feature edges, while edges B, C, and F would be ignored for edge-to-edge contact.

Output

The contact output variable CEDGEACTIVE is available to identify throughout the analysis if an edge is active as a primary edge, active as a secondary edge, or has been deactivated by the contact domain.

Surface Geometry Correction

By default, contact calculations are based on unsmoothed, faceted representations of the finite element surfaces in a general contact domain. Discrepancies between the true surface geometry and the faceted surface geometry might result in significant noise in the solution. Optional contact smoothing techniques simulate a more realistic representation of curved surfaces in the contact calculations. These techniques allow a discretized surface with discontinuous surface normals to more closely approximate the behavior of a smooth surface during an analysis. Improvements to results with the surface correction include more accurate contact stresses and less solution noise upon relative sliding between contact surfaces.

Contact smoothing can be specified for surfaces in a general contact domain using a surface property assignment. A single surface property assignment specifies all of the surfaces to be smoothed, as well as the appropriate geometry correction method for each surface. Three geometry correction methods can be employed:

- The circumferential smoothing method is applicable to surfaces approximating a portion of a surface of revolution.
- The spherical smoothing method is applicable to surfaces approximating a portion of a sphere.
- The toroidal smoothing method is applicable to surfaces approximating a portion of a torus (i.e., a circular arc revolved about an axis).

For each surface, you must specify the appropriate geometry correction method and either the approximate axis of revolution (for circumferential or toroidal smoothing) or the approximate spherical center (for spherical smoothing). For toroidal smoothing, you must also specify the distance of the center of the circular arc from the

axis of revolution. The center of the circular arc is then located such that the line it forms with point (X_a, Y_a, Z_a) is perpendicular with the axis of revolution.

Input File Usage: Use the following option to apply a geometric correction and define the parameters by specifying coordinates:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=GEOMETRIC CORRECTION, DEFINITION=COORDINATES
data lines to define smoothing regions

Several examples for data line entries are shown below. Repeat the data lines as many times as necessary to define the appropriate geometry corrections for all surfaces in the contact domain.

To apply circumferential smoothing to a surface with an axis of symmetry passing through points (X_a, Y_a, Z_a) and (X_b, Y_b, Z_b):

surface, CIRCUMFERENTIAL, X_a, Y_a, Z_a, X_b, Y_b, Z_b

To apply spherical smoothing to a surface with a spherical center at point (X_a, Y_a, Z_a):

surface, SPHERICAL, X_a, Y_a, Z_a

To apply toroidal smoothing to a surface with an axis of symmetry passing through points (X_a, Y_a, Z_a) and (X_b, Y_b, Z_b) with the center of the revolved circular arc at a distance *R* from the axis of symmetry:

surface, TOROIDAL, X_a, Y_a, Z_a, X_b, Y_b, Z_b, R

Use the following option to apply a geometric correction and define the parameters by specifying nodes:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=GEOMETRIC CORRECTION, DEFINITION=NODES
data lines to define smoothing regions

Several examples for data line entries are shown below. Repeat the data lines as many times as necessary to define the appropriate geometry corrections for all surfaces in the contact domain.

To apply circumferential smoothing to a surface with an axis of symmetry passing through nodes a and b:

surface, CIRCUMFERENTIAL, node a, node b

To apply spherical smoothing to a surface with a spherical center at node a:

surface, SPHERICAL, node a

To apply toroidal smoothing to a surface with an axis of symmetry passing through nodes a and b with the center of the revolved circular arc at a distance *R* from the axis of symmetry:

surface, TOROIDAL, node a, node b, R

Abaqus/CAE Usage: Contact surface smoothing can be applied only to native geometry models in Abaqus/CAE. Abaqus/CAE can automatically detect all circumferential, spherical, and toroidal surfaces in the general contact domain that can be smoothed and apply the appropriate smoothing.

Use the following option to enable automatic surface smoothing of a model:

Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Surface smoothing assignments: Edit**:
toggle on **Automatically assign smoothing for geometric faces**

Use the following option to manually apply smoothing to a surface:

Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Surface smoothing assignments: Edit**:
Select the surface, click the arrows to transfer the surface to the list of smoothing assignments.
In the **Smoothing Option** column, select **REVOLUTION** to apply circumferential smoothing, select **SPHERICAL** to apply spherical smoothing, select **TOROIDAL** to apply toroidal smoothing, or select **NONE** to prevent smoothing of the surface.

Considerations for Geometric Correction

The contact smoothing technique assumes that the initial locations of the surface nodes lie on the true initial surface geometry, with the exception of midedge nodes of C3D10M elements. This smoothing technique remains effective even if the midedge nodes of C3D10M elements do not lie on the true initial geometry (models meshed using Abaqus/CAE always have midedge nodes placed on the true initial geometry, but this might not be the case with other meshing preprocessors).

The effects of contact smoothing tend to be most significant for analyses involving small deformation, and the smoothing technique works well for cases involving large relative motion between the surfaces. For analyses with large deformation this smoothing technique typically has an insignificant effect on the solution. However, in some cases—especially where the underlying elements can fail—the smoothing can degrade the solution accuracy after large deformation.

By default, Abaqus/Explicit excludes contact edges on a surface with contact smoothing from the analysis. You can specify to include them; however, in this case the contact edges will conform to the faceted geometry, not to the smoothed surface.

Input File Usage: Use the following option to include contact edges on a surface with contact smoothing in the analysis:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FEATURE EDGE CRITERIA

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments**: **Edit**

Effects of Geometric Correction

The impact of contact surface smoothing can be demonstrated by a simple model of contact between concentric cylinders with a small clearance between them. With a matched mesh as shown in *Figure 9* there are no initial overclosures; therefore, there are no initial strain-free initial displacement adjustments. However, if the inner cylinder is rotated, the cylinders develop stresses (see *Figure 10*) as contact is detected due to the linear faceted

representation of the main surface. This behavior is improved when the circumferential smoothing technique is applied to the contacting surfaces of the two cylinders.

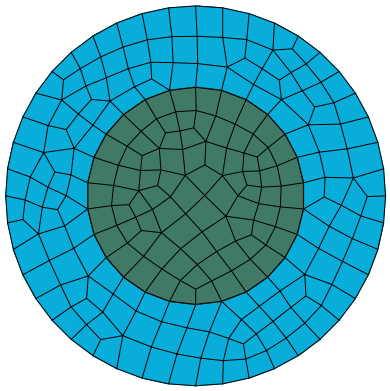


Figure 9: Concentric cylinders with matched mesh.

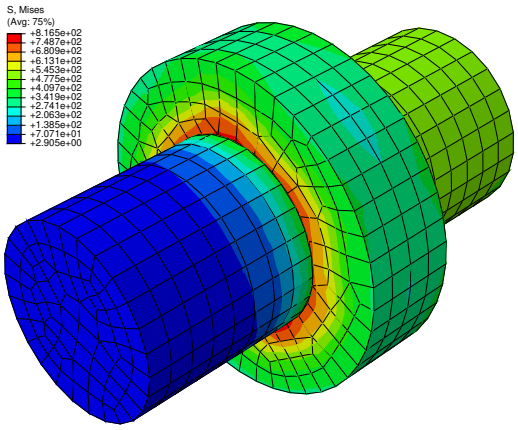


Figure 10: Stresses as cylinder rotates.

Surface-Based Friction Coefficients

In Abaqus/Explicit you can establish friction coefficients as mathematical combinations of coefficients specified as surface properties (see *Deriving Friction Coefficients from Quantities Specified as Surface Properties*). For contact between surfaces with identical surface-based coefficients, the function to compute the friction coefficient for an interface returns the same coefficient; otherwise, this function returns a coefficient between the two surface-based coefficients and closer to the lower of the surface-based coefficients. See *Deriving Friction Coefficients from Quantities Specified as Surface Properties* for more details about this capability, including user control of the function for computing interaction friction coefficients for surface-based friction coefficients.

Input File Usage: Use the following option to assign surface-based friction coefficients:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=FRICTION
surface or *material*, *friction_coefficient*, SURFACE (default) or MATERIAL

The third entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Surface friction assignments: Edit**:
Select surface or material, click the arrows to transfer surface or material to list of friction
assignments, and enter a value for the friction coefficient in the **Friction Coefficient** column.

Orientations

For surface regions, you can specify

- the initial orientation of local tangent directions and/or
- the degree of frictional directional preference for the local t_1 versus t_2 tangent directions in the context of an anisotropic friction model.

For each surface region, you can refer to a named orientation system and, if desired, an extra rotation (in degrees) applied to the orientation system once it has been projected to the surface. If no orientations are specified or an analytical rigid surface is used, Abaqus initializes the contact directions using the standard convention (see *Conventions*).

The specified local coordinate system is associated with a surface; whereas, the local tangent directions discussed in *Local Tangent Directions for Contact* are associated with contact constraints. The local coordinate system for contact is inherited from one of the surfaces, as discussed in *Local Tangent Directions for Contact*.

A preferred frictional direction for a surface in conjunction with anisotropic friction behavior can be specified using a frictional directional preference factor ϵ (default) or a frictional directional preference ratio r (see *Anisotropic Friction with Directional Preference as a Surface Property*).

Input File Usage: Use the following option to define a preferred contact orientation:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=ORIENTATION
surface or *material*, *orientation_system*, *extra_rotation_angle*,, SURFACE (default) or MATERIAL

The fifth entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Use the following option to define the frictional directional preference factor ϵ:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=ORIENTATION, FRICTION ANISOTROPY=EPSILON
, , , ϵ

Use the following option to define the ratio of the friction coefficients r:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=ORIENTATION, FRICTION ANISOTROPY=RATIO
, , , r

Abaqus/CAE Usage: Defining a preferred contact orientation, frictional directional preference factor, and ratio of friction coefficients is not supported in Abaqus/CAE.

Preferred Fraction of Frictional Work Directed to a Surface as a Surface Property

In Abaqus/Explicit you can specify the preferred fraction of frictional work of an interaction directed to a surface as a surface property. The default fraction is 0.5, which directs half of the friction work of an interaction to each surface. If the preferred fractions of the surfaces in an interaction do not sum to unity, a normalization process occurs in the context of the interaction such that the actual frictional work distribution fractions for that interaction sum to unity. This normalization process is described by the following equations:

$$f_{surfA}^{AB} = \begin{cases} \frac{1}{2} \text{ if } f_{A_pref} = f_{B_pref} = 0 \\ \frac{f_{A_pref}}{f_{A_pref}+f_{B_pref}} \text{ otherwise} \end{cases}, \; f_{surfB}^{AB} = 1 - f_{surfA}^{AB}.$$

Frictional work distribution factors influence the nodal frictional work output but have no influence on the distribution of heat associated with friction to the respective interacting surfaces, which can be influenced with pre-existing gap heat generation controls (*Modeling Heat Generated by Nonthermal Surface Interactions*).

Input File Usage: Use the following option to assign surface-based friction work distributions:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=DISTRIBUTION FACTOR
surface or *material*, *preferred fraction (between 0.0 and 1.0)*, SURFACE (default) or MATERIAL

The third entry indicates whether the first entry refers to a surface or material name. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Assigning preferred fractions of frictional work is not supported in Abaqus/CAE.

Assigning Contact Properties for General Contact in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *About General Contact in Abaqus/Explicit*
- *About Mechanical Contact Properties*
- *Contact Pressure-Overclosure Relationships*
- *Contact Damping*
- *Frictional Behavior*
- **CONTACT*
- **CONTACT PROPERTY ASSIGNMENT*
- **SURFACE INTERACTION*
- *Specifying and modifying contact property assignments for general contact*

Overview

Contact properties:

- define the mechanical surface interaction models that govern the behavior of surfaces when they are in contact; and
- can be applied selectively to particular regions within a general contact domain.

Assigning Contact Properties

The default contact property model in Abaqus/Explicit assumes "hard" contact in the normal direction, no friction, no thermal interactions, etc. You can assign a nondefault contact property definition (surface interaction) to specified regions of the general contact domain. These regions are identified with surface names or material names. For example, contact property `ContProp_A` can be assigned to the combination of surface `Surf_1` and the surface whose underlying elements have a section assignment with material `Rubber`. Surfaces associated with `Surf_1` and `Rubber` can span multiple parts.

Contact property assignments propagate through all analysis steps in which the general contact interaction is active.

The surface names used to specify the regions where nondefault contact properties should be assigned do not have to correspond to the surface names used to specify the general contact domain. In many cases the contact interaction will be defined for a large domain, while nondefault contact properties will be assigned to a subset of this domain. Any contact property assignments for regions that fall outside of the general contact domain will be ignored. The last assignment will take precedence if the specified regions overlap.

Input File Usage: **CONTACT PROPERTY ASSIGNMENT*
surface_1 or *material_1*, *surface_2* or *material_2*, *interaction_property_name*, SURFACE (default) or MATERIAL, SURFACE (default) or MATERIAL

This option must be used in conjunction with the **CONTACT* option. It should appear at most once per step; the data line can be repeated as often as necessary to assign contact properties to different regions.

If the first entry is omitted, a default surface that encompasses the entire general contact domain is assumed.

If the second entry is omitted or is the same as the first entry, contact between the first named or implied surface and itself is assumed. Keep in mind that surfaces can span multiple unattached bodies, so self-contact of a surface is not limited to contact of a single body with itself.

If the interaction property name is omitted, the unnamed set of default contact properties in Abaqus/Explicit is assumed. If an interaction property name is specified, it must also appear as the value of the NAME parameter on a **SURFACE INTERACTION* option in the model portion of the input file.

The fourth and fifth entries indicate whether the first and second entries, respectively, refer to surface or material names. If omitted, Abaqus assumes that a surface name is used.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**:
Contact Properties:
Individual property assignments: Edit: select the combinations of surfaces and materials and the contact
property in the columns on the left, and click the arrows in the middle to transfer them to the list of contact property assignments
or
Global property assignment: *interaction_property_name*

In Abaqus/CAE you must assign a contact property definition to every general contact interaction; Abaqus/CAE does not assume a default contact interaction property.

Example

The following contact property assignments are specified below for the first step in a general contact analysis:

- a global assignment of `contProp1` to the entire general contact domain;
- a local assignment of `contProp2` to self-contact for `surf1`;
- a local assignment of the default Abaqus contact property to contact between `surf2` and `surf3`; and
- a local assignment of `contProp3` to contact between the entire contact domain and `surf4`.

```
*SURFACE INTERACTION, NAME=contProp1
*FRICTION
0.1
*SURFACE INTERACTION, NAME=contProp2
*FRICTION
0.15
*SURFACE INTERACTION, NAME=contProp3
*FRICTION
0.20
*STEP
Step1
*DYNAMIC, EXPLICIT
…
*CONTACT
*CONTACT INCLUSIONS, ALL EXTERIOR
*CONTACT PROPERTY ASSIGNMENT
 , , contProp1
```

```
surf1, surf1, contProp2
surf2, surf3,
 , surf4, contProp3
```

Changing Contact Properties

Contact property models for general contact interactions are independent of the steps in which they are used and cannot be modified from step to step. To change the contact properties used in a given step, you must specify a new contact property assignment that refers to a different contact property model.

Example

For example, the following input could be used to change the friction coefficient used for contact between the entire general contact domain and `surf4` in the second step of the analysis started in the previous example:

```
*STEP
Step2
*DYNAMIC, EXPLICIT
…
*CONTACT
*CONTACT PROPERTY ASSIGNMENT
 , surf4, contProp2
```

Defining Contact Pairs in Abaqus/Explicit

In this section:

- *About Contact Pairs in Abaqus/Explicit*
- *Assigning Surface Properties for Contact Pairs in Abaqus/Explicit*
- *Assigning Contact Properties for Contact Pairs in Abaqus/Explicit*

About Contact Pairs in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *Element-Based Surface Definition*
- *Node-Based Surface Definition*
- *Analytical Rigid Surface Definition*
- *About Contact Interactions*
- **CONTACT CONTROLS*
- **CONTACT PAIR*
- **SURFACE*
- *Defining surface-to-surface contact*
- *Defining self-contact*

Overview

Abaqus/Explicit provides two algorithms for modeling contact and interaction problems: the general contact algorithm and the contact pair algorithm. See *About Contact Interactions* for a comparison of the two algorithms. This section describes how to define contact pairs with surfaces for contact simulations in Abaqus/Explicit.

Contact pairs in Abaqus/Explicit:

- are part of the history definition of the model and can be created, modified, and removed from step to step (unlike Abaqus/Standard, where contact pairs are model data);
- use sophisticated tracking algorithms to ensure that proper contact conditions are enforced efficiently;
- can be used simultaneously with the general contact algorithm (that is, some interactions can be modeled with contact pairs, while others are modeled with the general contact algorithm);
- can be formed using a pair of rigid or deformable surfaces or a single deformable surface;
- do not require surfaces with matching meshes;
- cannot be formed with one two-dimensional surface and one three-dimensional surface; and
- cannot be used for self-contact where the surface is composed of both first-order elements and second-order elements.

Defining a Contact Pair Interaction

The definition of a contact pair interaction in Abaqus/Explicit consists of specifying:

- the contact pair algorithm and the surfaces that interact with one another, as described in this section;
- the contact surface properties (*Assigning Surface Properties for Contact Pairs in Abaqus/Explicit*);
- the mechanical contact property models (*Assigning Contact Properties for Contact Pairs in Abaqus/Explicit*);
- the contact formulation (*Contact Formulations for Contact Pairs in Abaqus/Explicit*);
- the contact constraint enforcement method (*Contact Constraint Enforcement Methods in Abaqus/Explicit*); and
- the algorithmic contact controls (*Common Difficulties Associated with Contact Modeling Using Contact Pairs in Abaqus/Explicit*).

Defining a Contact Pair Containing Two Surfaces

To define a contact pair, you must indicate which pairs of surfaces will interact with each other. The order in which the surfaces are specified is important only when a nondefault weighting factor is specified (see *Contact Surface Weighting* for details). See *Element-Based Surface Definition*, *Node-Based Surface Definition*, and *Analytical Rigid Surface Definition* for information on defining surfaces for use in contact pairs.

Input File Usage: **CONTACT PAIR*
surface_1_name, *surface_2_name*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface-to-surface contact (Explicit)**: select the first surface, click **Surface**, select the second surface

Defining Self-Contact

Define contact between a single surface and itself by specifying only a single surface or by specifying the same surface twice.

Input File Usage: Use either of the following options:

**CONTACT PAIR*
surface_1,
**CONTACT PAIR*
surface_1, surface_1

Abaqus/CAE Usage: Interaction module: **Create Interaction**:
Self-contact (Explicit): select the surface
or
Surface-to-surface contact (Explicit): select the surface, click **Surface**, select the surface again

Limitations with Self-Contact

The following limitations are enforced for a contact pair with self-contact:

- The balanced main-secondary contact algorithm will always be used for the contact pair (a nondefault weighting factor cannot be specified for the contact pair).
- A contact thickness must be considered for self-contact surfaces on shell or membrane elements (see *Element-Based Surface Definition*); that is, a zero surface thickness (see *Forcing Zero Surface Thickness and Offset*) causes Abaqus/Explicit to issue an error message. By default, the contact thickness is equal to the current thickness.
- The contact thickness for self-contact should not exceed the edge lengths or diagonal lengths of the facets. You can reduce the contact thickness, if necessary; see *Controlling the Effects of Surface Thickness and Offset in Contact Calculations*.
- A specialized finite-sliding tracking algorithm must be used. The use of the small-sliding contact formulation is not supported and causes Abaqus/Explicit to issue an error message.
- Contact will be recognized between any node on a self-contact surface and any other point on the same surface, including either side of shells or membranes (that is, self-contact on shells and membranes is independent of the face identifier specified in the surface definition).

Removing and Adding Contact Pairs

Removal and addition of contact pairs:

- can be used to simulate complicated forming processes where multiple tools need to interact with the workpiece at different stages;
- can be used to extend surfaces to prevent one surface from sliding off another;
- can result in significant computational savings by eliminating unnecessary contact searches; and
- can be used to change the definition of a contact pair.

Adding Contact Pairs

By default, the contact pairs specified are added to the list of active contact pairs in the model.

Initial penetrations should be avoided for contact pairs introduced after the first step, as large nodal accelerations and severe element distortions can result (see *Contact Initialization for Contact Pairs in Abaqus/Explicit*). Redefining a contact pair by deleting it and adding it in the same step can also lead to problems, because the "state" information associated with the secondary nodes in contact will be reinitialized. For example, a penalty contact secondary node with a penetration past the midsurface of a double-sided main surface would be allowed to pass through the main surface if the contact state were reinitialized.

Input File Usage: **CONTACT PAIR*, OP=ADD

Abaqus/CAE Usage: Interaction module: **Create Interaction**

Removing Contact Pairs

Removal of contact pairs is a useful technique for simulating complicated forming processes where multiple tools will contact the same workpiece. Removing a contact pair once it is no longer needed eliminates the need to monitor the contact conditions and reduces the cost of the simulation.

Input File Usage: **CONTACT PAIR*, OP=DELETE

Abaqus/CAE Usage: Interaction module: interaction manager: **Deactivate**

General Restrictions on Surfaces Used in Contact Pairs

The following general restrictions (in addition to those discussed in *Element-Based Surface Definition*) apply to all surfaces used in contact pairs:

- The surface normals of a surface must point toward the other surface that it may contact except when the surface is double-sided, as discussed below.
- Element-based surfaces should not be used in contact pairs if the underlying elements may fail (see *Dynamic Failure Models* for more information). Use general contact (*About General Contact in Abaqus/Explicit*) or node-based surfaces (*Node-Based Surface Definition*) in such cases.
- The surface must be continuous, as discussed below.
- Continuum and structural elements cannot be mixed in the same surface definition.
- The surface cannot include C3D10 elements as underlying elements. Use general contact (*About General Contact in Abaqus/Explicit*) in such cases.

- Deformable elements cannot be combined with elements that are part of a rigid body to define a single surface.
- Surface trimming does not apply to contact pairs in Abaqus/Explicit (for more information,refer to the **SURFACE*, TRIM parameter in *Optional parameters*).

These restrictions do not apply to surfaces used with the general contact algorithm (*About General Contact in Abaqus/Explicit*).

The following restrictions apply to the surfaces forming a kinematic contact pair:

- Rigid surfaces must always be the main surface.
- Secondary surfaces must be part of a deformable body.
- A node-based surface can be used only as a secondary surface.

The following restrictions apply to the surfaces forming a penalty contact pair:

- Analytical rigid surfaces must always be the main surface.
- A node-based surface can be used only as a secondary surface.

Orienting the Surface's Normal

The orientation of a surface's normal can be critical for the proper detection of contact between two contacting surfaces. At the point of closest proximity the normals of a single-sided main surface forming the contact pair should always point toward the secondary surface. If, in the initial configuration of the model, a single-sided main surface's normal points away from its secondary surface, Abaqus/Explicit will detect that the secondary surface penetrates the main surface. Abaqus/Explicit will attempt to resolve this initial overclosure of the contact pair with strain-free displacements before the start of the simulation (see *Contact Initialization for Contact Pairs in Abaqus/Explicit*). Abaqus/Explicit may have difficulty with the simulation if the overclosure is too severe. In most of these cases the analysis will terminate immediately, and an error message about severely distorted elements will be issued.

You must give particular attention to checking that analytical rigid surfaces or single-sided surfaces created on shell, membrane, or rigid elements have the proper orientation. Surface orientation errors can often be quickly and easily detected by running a data check analysis (*Abaqus/Standard and Abaqus/Explicit Execution*) and inspecting the deformed configuration in Abaqus/CAE. If large displacements have occurred, they may be due to an incorrect surface orientation.

The proper and improper orientation of a rigid and deformable surface is shown in *Figure 1*.

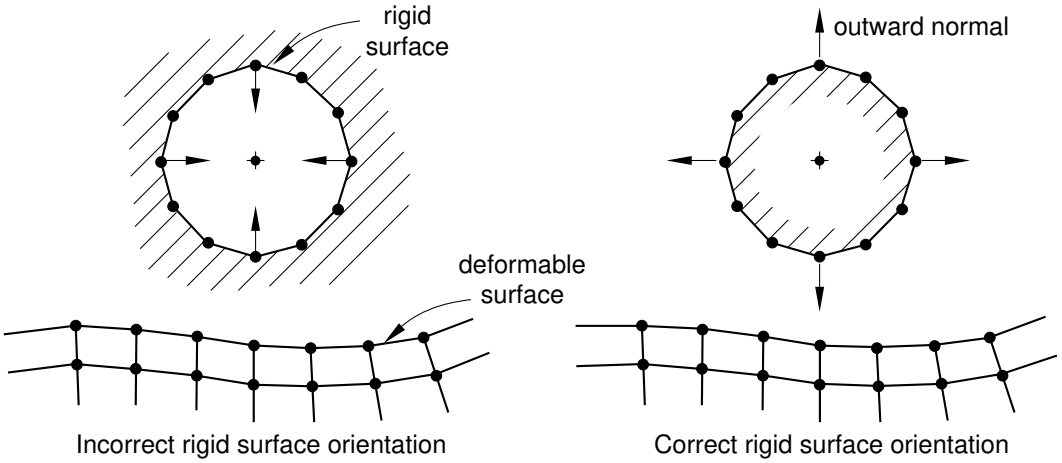


Figure 1: Example of proper and improper surface orientation with a rigid surface.

It is not necessary for the normals of all of the underlying shell or membrane elements to have a consistent positive orientation for a double-sided surface: if possible, Abaqus/Explicit will define the surface such that its facets have consistent normals, even if the underlying elements do not have consistent normals. The facet normals will be the same as the element normals if the element normals are all consistent; otherwise, an arbitrary positive orientation is chosen for the surface. For double-sided surfaces the positive orientation is significant only with respect to the sign of the contact pressure output variable, CPRESS, as discussed in *Element-Based Surface Definition*.

Defining a Continuous Surface

A contact pair surface cannot be made up of two or more disconnected regions. The definition of analytical rigid surfaces automatically ensures that these surfaces are continuous. However, care must be taken to define surfaces formed with elements so that they are continuous across element edges in three-dimensional models or through nodes in two-dimensional models. This continuity requirement has several implications for what constitutes a valid or invalid surface definition. In two dimensions the surface must be either a simple, nonintersecting curve with two terminal ends or a closed loop. *Figure 2* shows examples of valid and invalid two-dimensional surfaces for use in contact pairs.

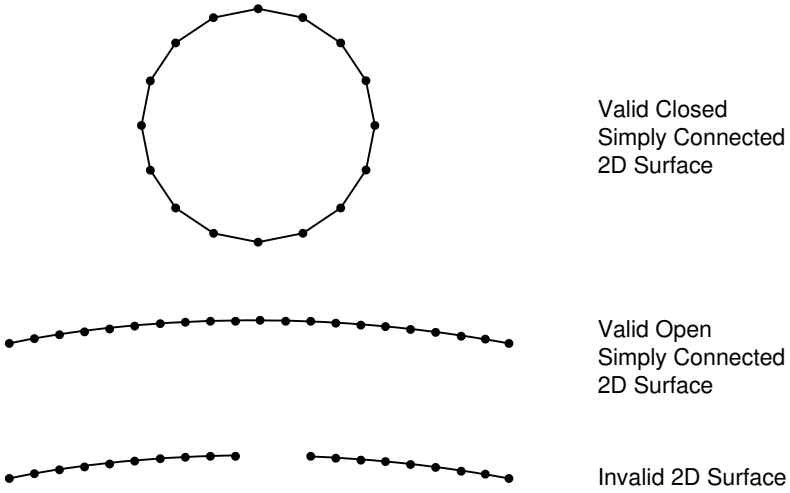
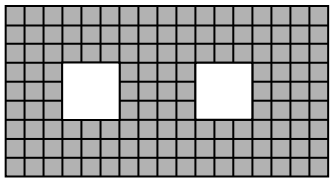
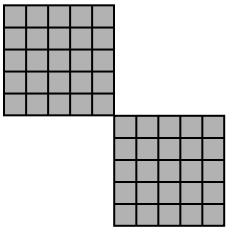


Figure 2: Valid and invalid 2D surfaces.

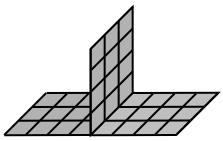
In three dimensions an edge of an element face belonging to a valid surface may be either on the perimeter of the surface or shared by one other face. Two element faces forming a contact pair surface cannot be joined just at a shared node; they must be joined across a common element edge. An element edge cannot be shared by more than two surface facets. *Figure 3* illustrates valid and invalid three-dimensional surfaces for use in contact pairs.



Valid Simply Connected Surface



Invalid Surface



Invalid Surface

Figure 3: Valid and invalid 3D surfaces.

The continuity requirement applies to both automatically generated free surfaces and surfaces defined with element face identifiers (see *Element-Based Surface Definition*). *Figure 4* shows an automatically generated free surface resulting from the specification of an element set consisting of two disjointed groups of elements. The resulting surface is not continuous since it is composed of two disjoint open curves.

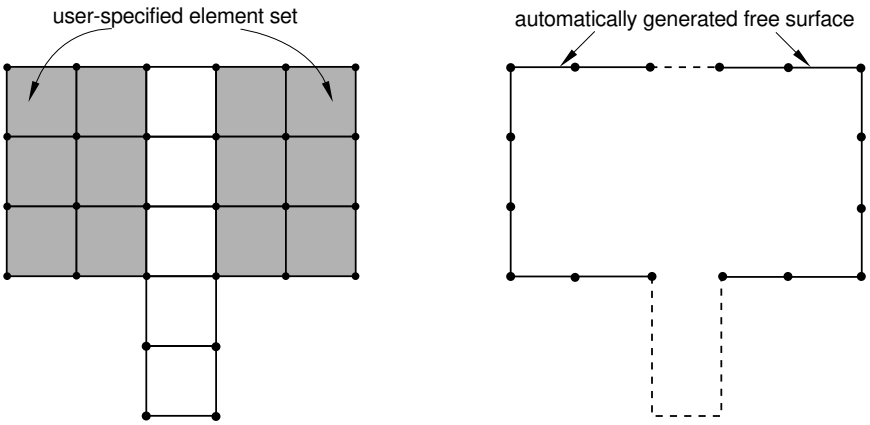


Figure 4: Automatic free surface generation.

Restrictions for Two-Dimensional Contact Simulations

The following restrictions apply when defining a contact simulation for two-dimensional (planar) or axisymmetric problems:

- A contact pair cannot involve a planar surface and an axisymmetric surface. This restriction applies only to deformable and element-based rigid surfaces.
- Defining a contact pair that contains two surfaces formed by planar elements of different sizes in the out-of-plane direction (“depth”) is not recommended and will result in a warning message. In such a case frictional stresses are calculated based on a weighted average depth, with the weighting for the first surface equal to the user-specified contact surface weighting factor. The out-of-plane thickness for two-dimensional

beam element-based surfaces is always assumed to be one. As a result, the contact pressure acting on such a surface can be considered as a line force as well.

- When more than one contact pair involves contact between the same rigid surface formed by planar elements and different planar deforming surfaces, the deforming surfaces must all have the same depth; otherwise, a warning message will be issued. The depth value used for calculating contact stresses will then be taken from one of these deforming surfaces, but this choice cannot be predicted.

Limitations in Contact Simulations with Three-Dimensional Beam and Truss Elements

Element-based surfaces cannot be formed on three-dimensional beam or truss elements, so node-based surfaces must be used to define a surface on these elements. Because a node-based surface must be used, a surface on three-dimensional beam or truss elements must always form the secondary surface in a pure main-secondary contact pair. Therefore, it is not possible to have two three-dimensional beam or truss structures contact each other.

Output

You can write the contact surface variables associated with the interaction of contact pairs to the Abaqus output database (`.odb`) file. The surface variables for a mechanical contact analysis include contact pressure and force, frictional shear stress and force, relative tangential motion (slip) of the surfaces during contact, whole surface resultant quantities (that is, force, moment, center of pressure, and total area in contact), the status of bonded nodes, and the maximum torque transmitted about the z-axis of axisymmetric elements.

Additional discussion on requesting contact surface output can be found in *Writing Surface Output to the Output Database*. Output from thermal interactions is discussed in *Thermal Contact Properties*.

Field Output

The generic variables CSTRESS, CFORCE, FSLIP, and FSLIPR are valid field output requests for Abaqus/Explicit. If CSTRESS is requested for a contact pair, the variables CPRESS (contact pressure), CSHEAR1 (contact traction in the local 1-direction), and, if the contact interaction is three-dimensional, CSHEAR2 (contact traction in the local 2-direction) can be contoured in Abaqus/CAE for each discrete (that is, non-analytical) surface in a contact pair.

Contours of contact pressure (CPRESS) on surfaces used with the contact pair algorithm will be displayed using the convention that a positive pressure represents compressive contact on the positive side of the surface. The positive side of the surface can be determined by drawing the surface normals in the Visualization module of Abaqus/CAE. Following this convention, the sign of CPRESS will be reversed for contact on the negative (back) side of a double-sided surface, so negative values of CPRESS may be seen if contact occurs on the back side of a double-sided surface. If contact from separate contact pairs occurs on both sides of the double-sided surface at the same point, the value of CPRESS is given for each contact pair separately.

If CFORCE is requested for a contact pair, the variables CNORMF (normal contact force) and CSHEARF (shear contact force) can be plotted as vectors in a symbol plot in Abaqus/CAE for each discrete (that is, non-analytical) surface in a contact pair.

If FSLIPR is requested, FSLIPR (the magnitude of the slip rate for secondary nodes in contact) can be contoured in Abaqus/CAE for each secondary surface in a contact pair. In addition, for three-dimensional contact interactions involving an analytical rigid surface and for all two-dimensional contact interactions, components of net slip rate based on local tangent directions (FSLIPR1 and, in three dimensions, FSLIPR2) can also be contoured in Abaqus/CAE for each secondary surface in a contact pair if FSLIPR is requested. All of the slip rate variables associated with FSLIPR are zero whenever a secondary node is not in contact.

If FSLIP is requested, FSLIPEQ (the length of the overall slip path for a secondary node while it is in contact) can be contoured in Abaqus/CAE for each secondary surface in a contact pair. In addition, for three-dimensional

contact interactions involving an analytical rigid surface and for all two-dimensional contact interactions, components of net slip (FSLIP1 and, in three dimensions, FSLIP2) can also be contoured in Abaqus/CAE for each secondary surface in a contact pair if FSLIP is requested. These slip variables are equivalent to the slip rate variables integrated over time: FSLIPEQ, FSLIP1, and FSLIP2 are equivalent to FSLIPR, FSLIPR1, and FSLIPR2 integrated over time, respectively. Therefore, these slip variables account only for relative motions that occur while secondary nodes are in contact.

The algorithm used to establish and evolve local tangent directions for contact pairs is described in *Local Tangent Directions for Contact*. Unlike general contact, previously accumulated slip components for contact pairs, FSLIP1 and FSLIP2, are not resolved into the new local system before incremental contributions are added to them.

Displacement field output (U) for the entire model is written to the output database automatically when any of the contact field output variables are requested.

History Output

Several whole surface contact variables are available as history output. These variables record the contact state of a surface as a set of force (CFN, CFS, and CFT) and moment (CMN, CMS, and CMT) resultants with respect to the origin. Additional variables give the center of pressure (XN, XS, and XT) on the surface (defined as the point closest to the centroid of the surface that lies on the line of action of the resultant force for which the resultant moment is minimal). The last letter of each variable name (except the variable CAREA) denotes which contact force distribution on the surface is used to calculate the resultant: the letter N denotes that the normal contact forces are used to derive the resultant quantity; the letter S denotes that the shear contact forces are used to derive the resultant quantity; and the letter T denotes that the sum of the normal and shear contact forces are used to derive the resultant quantity. These history output variables will be written twice to the output database once for each surface involved in the contact pair.

Each total moment output variable will not necessarily equal the cross product of the respective center of force vector and resultant force vector. Forces acting on two different nodes of a surface may have components acting in opposite directions, such that these nodal force components generate a net moment but not a net force; therefore, the total moment may not arise entirely from the resultant force. The center of force output variables tend to be most meaningful when the surface nodal forces act in approximately the same direction.

The total area in contact at a given time can be requested using output variable CAREA, defined as the sum of all the facets where there is contact force. The contact area reported by CAREA is generally slightly larger than the true contact area for reasonably meshed contact surfaces; therefore, interpretation of CAREA should be done with care. The discrepancy between the CAREA output and the true contact area decreases as the mesh density increases. Using contact inclusions or exclusions to limit CAREA output to smaller contact surfaces may also reduce the discrepancy in some cases. Since the CAREA output is an approximation of the true contact area, deriving force or stress values using this output may not yield accurate values; requesting contact force and stress directly is the most appropriate way to obtain accurate results.

Obtaining the "maximum torque" That Can Be Transmitted about the z-Axis in an Axisymmetric Analysis

When modeling surface-based contact with axisymmetric (CAX) elements, Abaqus/Explicit can calculate the maximum torque (output variable CTRQ) that can be transmitted about the z-axis. The maximum torque, T, is defined as

$$T = \int\int r^2 p \, ds \, d\theta,$$

where p is the pressure transmitted across the interface, r is the radius to a point on the interface, and s is the current distance along the interface in the r–z plane. This definition of "torque" effectively assumes a friction coefficient of unity.

Assigning Surface Properties for Contact Pairs in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *About Contact Pairs in Abaqus/Explicit*
- **CONTACT PAIR*
- **SURFACE*
- *Specifying geometric properties for mechanical contact property options*

Overview

This section describes how to modify the surface properties for contact interactions in Abaqus/Explicit defined with the contact pair algorithm, including the surface thickness and offset.

Shell, Membrane, or Rigid Element Thickness and Shell or Rigid Element Offset

To define surfaces on shell, membrane, or rigid elements such that they are in contact at the start of the analysis, the element thicknesses must be considered when defining the nodal coordinates; otherwise, the surfaces in the contact pair will be overclosed. Surface thickness and surface offset are properties that are inherited from underlying shell and membrane elements by default. For a surface based on rigid elements, the default surface thickness and offset correspond to the thickness and offset defined for the rigid body to which the elements belong (see *Rigid Elements*). The surface thickness and offset are zero for surfaces based on solid elements.

By default, the nodal thickness for surfaces based on shell, membrane, or rigid elements equals the minimum thickness of the surrounding elements (see *Figure 1* and *Table 1*). The surface thickness within a facet is interpolated from the nodal values; the interpolated surface thickness never extends past the specified element or nodal thickness, which may be significant with respect to initial overclosures.

If a spatially varying nodal thickness is defined for the underlying elements (see *Nodal Thicknesses*), the nodal surface thickness may not correspond exactly to the specified nodal thickness (see node 4 in *Figure 2* and *Table 2*). The nodal surface thickness distribution will tend to be more diffuse than the specified nodal thickness distribution (because the specified nodal thicknesses are averaged to compute the element thicknesses, and the minimum of the surrounding element thicknesses is the nodal surface thickness).

Effects of surface thickness and offsets, as well as methods for modifying the surface thickness and for avoiding surface offsets, are discussed below.

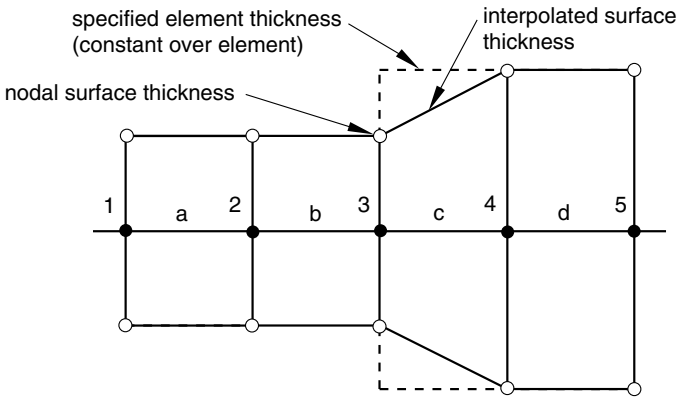


Figure 1: Continuous variation of surface thickness across facet boundaries.

Table 1: Thicknesses corresponding to figure showing continuous variation.

node	element	specified element thickness	nodal surface thickness (minimum of adjacent element thicknesses)
1			0.5
	a	0.5	
2			0.5
	b	0.5	
3			0.5
	c	0.9	
4			0.9
	d	0.9	
5			0.9

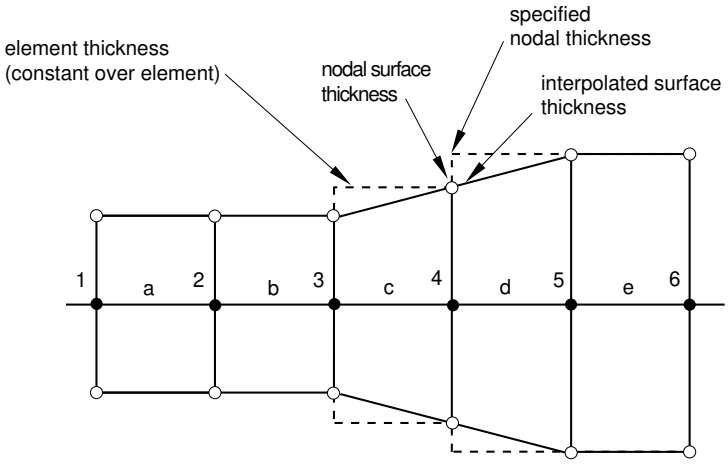


Figure 2: Small discrepancy between the nodal surface thickness and the specified nodal thickness.

Table 2: Thicknesses corresponding to figure showing small discrepancies.

node	element	specified nodal thickness	element thickness (average of specified nodal thickness)	nodal surface thickness (minimum of adjacent element thicknesses)
1		0.5		0.5
	a		0.5	
2		0.5		0.5
	b		0.5	
3		0.5		0.5
	c		0.7	
4		0.9		0.7
	d		0.9	
5		0.9		0.9
	e		0.9	
6		0.9		0.9

Effects of Surface Thickness and Offsets

Accounting for thickness in the contact pair algorithm will cause the surface to extend past the parent element boundary in the plane of the element by an amount equal to one-half its thickness. For example, this surface extension, which is semi-circular in shape, will cause contact to be established between the edge of a shell and an opposing surface before the node on the shell boundary reaches the opposing surface. The extension is present for both single-sided and double-sided surfaces. *Figure 3* demonstrates this concept. Such “bull-nose” extensions are avoided when the general contact algorithm (*About General Contact in Abaqus/Explicit*) is used. The effect of a shell or rigid offset on a surface is shown in *Figure 4*. Poorly defined surfaces can result near corners if large offsets are present, as shown in *Figure 5*. You should consider this when defining a model. A warning message will be issued if the offset magnitude is greater than one-half of any of the parent shell element edge lengths. However, at acute corners it is possible for an offset less than one-half of the parent element size to result in a poorly defined contact surface (and in this case no warning will be given).

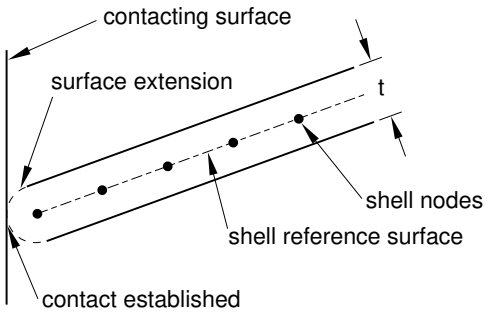


Figure 3: Extension of contact surface for edge contact without zero surface thickness.

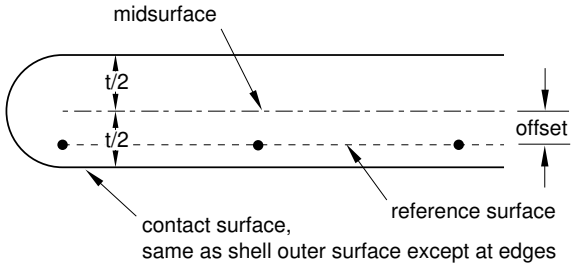


Figure 4: Extension of contact surface if a shell offset is present.

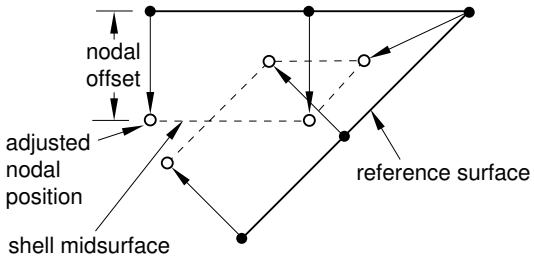


Figure 5: Example of a poorly defined surface near a corner when a large shell offset is present.

Controlling the Effects of Surface Thickness and Offset in Contact Calculations

You can control the thickness and offset used in the contact calculations only; they do not affect surface-based constraints. These settings are intended primarily for self-contact surfaces since you cannot force zero thickness for these surfaces, as described below.

Self-contact surfaces should not contain facets that are thicker than their edge or diagonal lengths. Extremely large thicknesses will cause nodes to appear to be penetrating nearby facets in even a flat self-contact surface due to the algorithmic use of a semi-circular tube with a radius of half the contact thickness around the edge of each facet (see *Figure 6*).

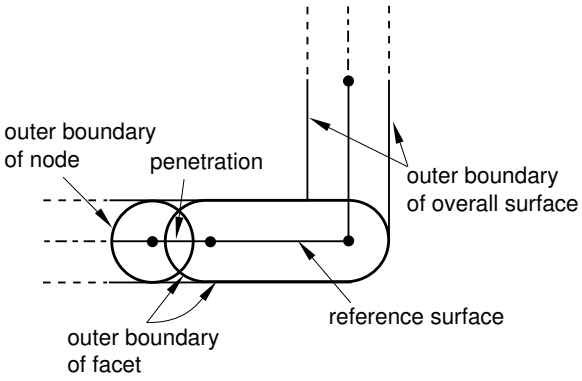


Figure 6: Undesired penetration resulting from a large thickness in a self-contact surface.

You can scale the effective thickness used for all of the facets on a surface by a single factor, f. Alternatively, you can adjust only the thicknesses for surface facets in which the thickness to minimum edge or diagonal length ratio exceeds a specified value, r; the amount by which a facet thickness is adjusted may vary during an analysis because of changes in the facet size. If the thickness to element size ratio exceeds 1.0 in the initial configuration for a self-contact surface, an error message recommending that you adjust the thickness will be issued.

You should not specify extremely small values for f or r for double-sided surfaces or surfaces that will be involved in self-contact since these surfaces must have a contact thickness that is significant compared to the facet size.

For surfaces involved only in two-surface contact it is acceptable to set *f*=0.0; however, it is computationally more efficient to use the method described below to force a zero surface thickness. It is also possible to enforce the offset but not the thickness in the surface model by setting the scale factor, *f*, equal to zero.

Input File Usage: Use the following option to scale the surface thickness by a single factor:

**SURFACE*, NAME=*name*, SCALE THICK=*f*

Use the following option to adjust the thickness to element size ratios:

**SURFACE*, NAME=*name*, MAX RATIO=*r*

Abaqus/CAE Usage: You cannot scale the thickness of a contact surface in Abaqus/CAE.

Forcing Zero Surface Thickness and Offset

You can force the surface thickness and offset to be zero, rather than inherit the thickness and offset of underlying shell, membrane, or rigid elements. In this case the contact surface is taken as the reference surface (see *Figure 7*).

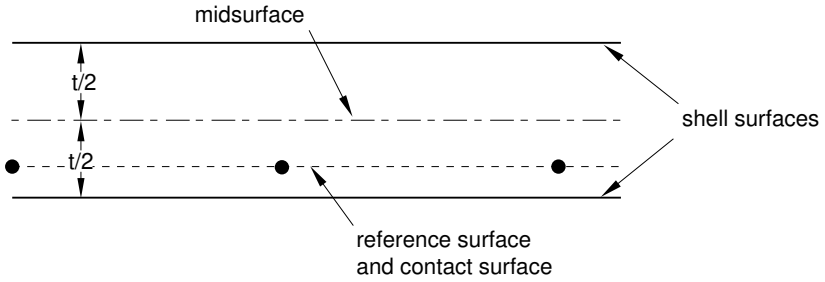


Figure 7: Contact surface with zero thickness and offset.

You cannot ignore the thickness for a surface that is used as a contact surface for single-surface (self) contact. If one of the surfaces in a contact pair is a double-sided surface, zero thickness can be forced on only one of the two surfaces: at least one surface in a contact pair involving double-sided surfaces must have a nonzero thickness. The ability to force zero surface thickness is useful for performing parameter studies on the thickness or offset of a model since you can change the thickness and offset without also having to move the mesh to control the initial separation between the surfaces.

Input File Usage: **SURFACE*, NAME=*name*, NO THICK

Abaqus/CAE Usage: You cannot force a surface thickness to be zero in Abaqus/CAE.

Example

Contact calculations are generally most accurate with the default treatment of thickness and offset. However, when a shell offset of half the original shell thickness has been specified, forcing zero surface thickness will give an accurate representation of one side of the surface. This approach can be more accurate near a corner (especially on the exterior side of a corner) than if the offset and thickness are enforced for the surface, as shown in *Figure 8*.

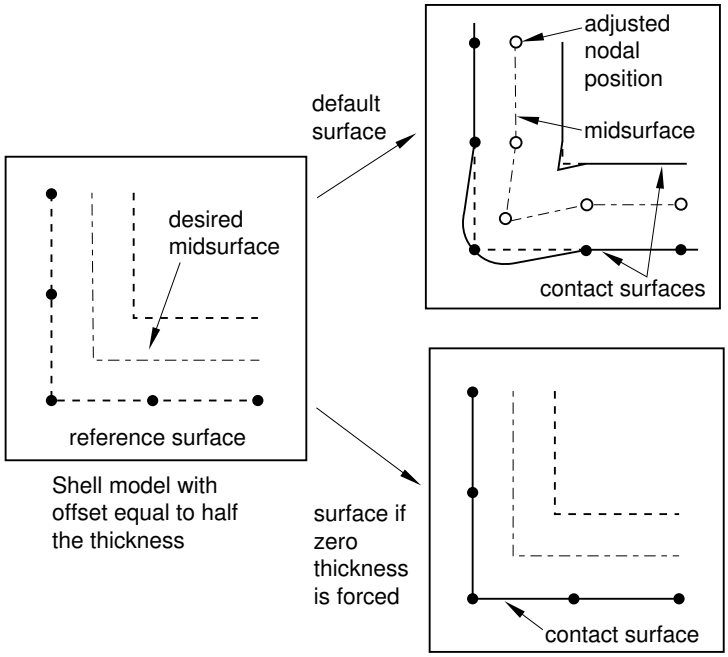


Figure 8: Forcing zero surface thickness when the shell offset is half the original shell thickness.

Forcing Zero Surface Offset

For situations in which it is desirable to ignore the effect of the offset but when it is still necessary to model the thickness in the contact calculations, you can force only the surface offset to be zero without affecting the surface thickness. In this case the contact surface is the outside surface of an imaginary shell, membrane, or rigid element whose midsurface is at the reference surface (see *Figure 9*).

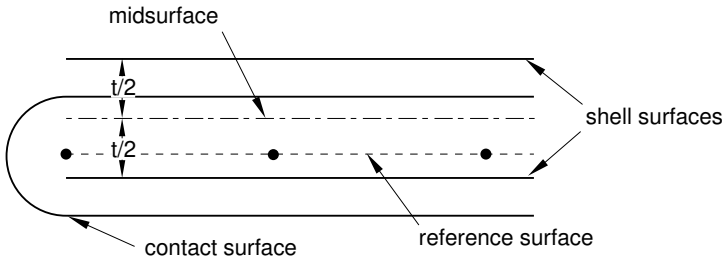


Figure 9: Contact surface with zero offset.

This method could be used for a self-contact surface that would be poorly defined if the offset were enforced (thickness must be enforced for self-contact surfaces).

Input File Usage: **SURFACE*, NAME=*name*, NO OFFSET

Abaqus/CAE Usage: You cannot force a surface offset to be zero in Abaqus/CAE.

Defining Additional Contact Thicknesses for a Contact Pair Interaction

You can specify a contact offset for a contact pair interaction in addition to any element thicknesses or midsurface offsets already defined for the elements underlying the contact pair surfaces. For small sliding this includes contact offsets defined by initial clearances (see *Specifying Initial Clearance Values Precisely*). The specified

offset value will be applied as an additional thickness of a layer separating the two surfaces, not as an additional thickness for each surface in the contact pair. This value can be positive or negative. This technique is often used in conjunction with softened behavior (see *Contact Pressure-Overclosure Relationships*) to model the thickness of a thin layer between two contacting surfaces.

Input File Usage: **SURFACE INTERACTION*, PAD THICKNESS=*value*

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Geometric Properties**: toggle on **Thickness of interfacial layer (Explicit):** *value*

Assigning Contact Properties for Contact Pairs in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *About Mechanical Contact Properties*
- *Contact Pressure-Overclosure Relationships*
- *Contact Damping*
- *Frictional Behavior*
- *User-Defined Interfacial Constitutive Behavior*
- **CONTACT PAIR*
- **SURFACE INTERACTION*
- *Interaction property editors*

Overview

Contact properties:

- define the mechanical and thermal surface interaction models that govern the behavior of surfaces when they are in contact; and
- are assigned to individual contact pairs.

Assigning a Contact Property Definition to a Contact Pair

If nondefault contact properties are desired, you can refer to a contact property definition that governs the interaction of the two surfaces.

Multiple contact pairs can refer to the same contact property definition.

Input File Usage: Use both of the following options:

**CONTACT PAIR*, INTERACTION=*interaction_property_name*
surface_1, *surface_2*
**SURFACE INTERACTION*, NAME=*interaction_property_name*

Abaqus/CAE Usage: Interaction module:

Create Interaction Property: **Name:** *interaction_property_name*, **Contact**

Interaction editor: **Contact interaction property**: *interaction_property_name*

Example

Figure 1 shows the mesh used in this example. For purposes of this example, a balanced main-secondary contact pair is used. The property definition for the contact pair (`GRATING`) uses a friction model where μ=0.4.

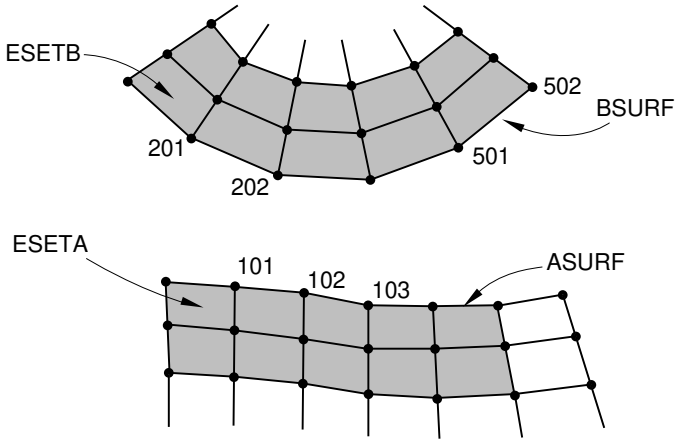


Figure 1: Surface interaction with friction.

```
*HEADING
…
*SURFACE, NAME=ASURF
ESETA,
*SURFACE, NAME=BSURF
ESETB,
…
*STEP
Step1
*DYNAMIC, EXPLICIT
…
*CONTACT PAIR, INTERACTION=GRATING
ASURF, BSURF
*SURFACE INTERACTION, NAME=GRATING
*FRICTION
0.4
```

Changing Contact Properties

Contact property models are defined as model or history data for a contact pair analysis. You can modify the contact properties from step to step; however, the old contact pair should be deleted and redefined using the new interaction.

Example

For example, the following input could be used to change the friction coefficient used for contact between `ASURF` and `BSURF` in the second step of the analysis started in the previous example:

```
*STEP
Step2
*DYNAMIC, EXPLICIT
…
*CONTACT PAIR, INTERACTION=GRATING,OP=DELETE
ASURF, BSURF
*SURFACE INTERACTION, NAME=GRATING_NEW
*FRICTION
0.5
*CONTACT PAIR, INTERACTION=GRATING_NEW
ASURF, BSURF
```

Contact Property Models

In this section:

- *Mechanical Contact Properties*
- *Thermal Contact Properties*
- *Electrical Contact Properties*
- *Pore Fluid Contact Properties*

Mechanical Contact Properties

In this section:

- *About Mechanical Contact Properties*
- *Contact Pressure-Overclosure Relationships*
- *Contact Damping*
- *Contact Blockage*
- *Frictional Behavior*
- *User-Defined Interfacial Constitutive Behavior*
- *Interaction of Debonded Surfaces*
- *Contact Cohesive Behavior*
- *Contact Wear*

About Mechanical Contact Properties

Overview

In a mechanical contact simulation the interaction between contacting bodies is defined by assigning a contact property model to a contact interaction (see *Contact Properties for General Contact in Abaqus/Standard*, *Assigning Contact Properties for Contact Pairs in Abaqus/Standard*, *Assigning Contact Properties for General Contact in Abaqus/Explicit*, and *Assigning Contact Properties for Contact Pairs in Abaqus/Explicit* for details). Mechanical contact property models:

- may include a constitutive model for the contact pressure-overclosure relationship that governs the motion of the surfaces;
- may include a damping model that defines forces resisting the relative motions of the contacting surfaces;
- may include a friction model that defines the force resisting the relative tangential motion of the surfaces;
- may include a constitutive model in which you define the normal and tangential behavior in user subroutine *UINTER* in Abaqus/Standard;
- may include a constitutive model in which you define the normal and tangential behavior in user subroutine *VUINTER* in Abaqus/Explicit when using the contact pair algorithm;
- may include a constitutive model in which you define the normal and tangential behavior in user subroutine *VUINTERACTION* in Abaqus/Explicit when using the general contact algorithm;
- in Abaqus/Standard may include a constitutive model for the penetration of fluid between two contacting surfaces;
- in Abaqus/Standard may include a constitutive model for the interaction of debonded surfaces;
- in Abaqus/Explicit may include a constitutive model that simulates the failure of bonds connecting the interacting bodies; and
- may include surface-based cohesive behavior that allows modeling of delamination of bonds or “sticky” contact using progressive damage evolution models.

This section provides a general outline of how to define the components of a mechanical contact property model. Specific details about the different components of the contact property models and the algorithms used for the contact calculations are found in other sections of this chapter.

Defining the Contact Property Model

There are different methods for defining the components of a mechanical contact property model.

Defining the Contact Pressure-Overclosure Relationship

The default contact pressure-overclosure relationship used by Abaqus is referred to as the “hard” contact model. Hard contact implies that:

- the surfaces transmit no contact pressure unless the nodes of the secondary surface contact the main surface;
- no penetration is allowed at each constraint location (depending on the constraint enforcement method used, this condition will either be strictly satisfied or approximated);
- there is no limit to the magnitude of contact pressure that can be transmitted when the surfaces are in contact.

You can define a nondefault pressure-overclosure relationship for a surface interaction. The various pressure-overclosure relationships available in Abaqus are discussed in *Contact Pressure-Overclosure*

Relationships, and the constraint methods available to enforce these relationships are discussed in *Contact Constraint Enforcement Methods in Abaqus/Standard*.

Defining a Friction Model

By default, Abaqus assumes that contact between surfaces is frictionless. You can include a friction model as part of a surface interaction definition.

Details of the various friction models available in Abaqus are discussed in *Frictional Behavior*.

Defining Contact Cohesive Behavior

You can define contact cohesive behavior to model delamination of initially bonded surfaces or to model "sticky" contact between parts that are initially separated but bond on coming into contact, with the possibility that the bond may undergo progressive damage and fail.

Contact cohesive behavior is modeled within the general contact framework in Abaqus/Explicit and within the general contact or contact pair framework in Abaqus/Standard. Details of the contact cohesive behavior model are discussed in *Contact Cohesive Behavior*.

Defining a Surface Interaction Model with Damping between the Surfaces

You can define damping forces to oppose the relative motion between the interacting surfaces.

In Abaqus/Standard the specified contact damping affects the motion in the normal direction only, whereas in Abaqus/Explicit contact damping can affect both the relative tangential motion and the motion normal to the surfaces.

The details of the contact damping model are discussed in *Contact Damping*.

Defining Contact Blockage in Abaqus/Explicit

In Abaqus/Explicit you can control the combination of surfaces that can cause blockage of flow out of a surface-based fluid cavity. The details of contact blockage are discussed in *Contact Blockage*.

User-Defined Interfacial Constitutive Behavior

Instead of choosing one or some combination of the various interfacial behavior models that are available in Abaqus, you can define any special or proprietary interfacial constitutive behavior through a user subroutine. In Abaqus/Standard you can use the subroutine *UINTER*; whereas in Abaqus/Explicit you can use *VUINTER* if you are using the contact pair algorithm and *VUINTERACTION* if you are using the general contact algorithm.

In Abaqus/Explicit a penalty enforcement of the contact constraint must be used for interacting surfaces whose interfacial behavior is governed by *VUINTER* or *VUINTERACTION*.

Details of the definition of a user-defined interfacial constitutive behavior are discussed in *User-Defined Interfacial Constitutive Behavior*.

Defining a Pressure Penetration Load in Abaqus/Standard

You can define pressure penetration loads to simulate the penetration of fluid between two contacting surfaces in Abaqus/Standard. The details of the pressure penetration model are discussed in *Fluid Pressure Penetration Loads*.

Defining CZone Crush Characteristics in Abaqus/Explicit

The CZone capability in Abaqus/Explicit integrates material, element, and contact modeling aspects to simulate crushing of laminated composites. This capability is discussed in *CZone Analysis*.

Defining the Interaction of Debonded Surfaces in Abaqus/Standard

You can allow two initially bonded surfaces to debond in Abaqus/Standard, as discussed in *Crack Propagation Analysis*. The details of the contact interaction model after debonding are discussed in *Interaction of Debonded Surfaces*.

Dependence of Contact Properties on Rates of Relative Motion

For cases in which contact properties depend on rates of relative motion, Abaqus/Explicit uses a filtered rate of relative motion to compute the corresponding contact property, as discussed in *Filtering Rates of Relative Motion*. If dependence of a contact property is based on interpolation of tabular data, Abaqus/Explicit uses logarithmic interpolation of rates of relative motion by default and linear interpolation with respect to other independent variables, as discussed in *Interpolating Tabular Data with Respect to Rates of Relative Motion*. Abaqus/Standard does not filter rates of relative motion and uses linear interpolation of all tabular data.

Filtering Rates of Relative Motion

Certain contact properties depend on slip rate or rate of separation. Abaqus/Explicit filters these rate measures to minimize the effects of high frequency noise; whereas Abaqus/Standard does not filter the rate measures. The formula used in Abaqus/Explicit to filter slip rate or rate of separation is

$$\dot{\bar{\delta}}_{curr} = \omega \frac{\Delta\delta}{\Delta t} + \left(1-\omega\right)\dot{\bar{\delta}}_{prev}.$$

Here, $\Delta\delta$ is the increment change in slip or equivalent separation over the time increment Δt,, and $\dot{\bar{\delta}}_{curr}$ and $\dot{\bar{\delta}}_{prev}$ are current and previous filtered values, respectively, of the equivalent rate of separation or rate of slip at a node. The factor $\omega, (0 < \omega \leq 1)$, facilitates filtering high-frequency oscillations associated with rate-dependent behavior. The default value of ω is 0.9. A value of $\omega = 1$ provides no filtering effect.

For discussion of the dependence of th friction coefficient on slip rate for both tabular dependence and for exponential decay friction behavior, see *Frictional Behavior*. For discussion of the dependence of cohesive damage initiation and evolution parameters on the effective rate of separation, see *Contact Cohesive Behavior*.

Input File Usage: Use the following option to define a constant filter factor:

**SURFACE INTERACTION*, RATE FACTOR=ω

Abaqus/CAE Usage: Defining a constant filter factor is not supported in Abaqus/CAE.

Interpolating Tabular Data with Respect to Rates of Relative Motion

Interpolation associated with tabular dependence of friction coefficients and cohesive damage parameters in Abaqus/Explicit can be logarithmic or linear with respect to rate dependencies. Abaqus/Standard always uses linear interpolation for tabular dependence.

The formula for logarithmic interpolation to determine the value of a given parameter ρ for a filtered slip/separation rate $\dot{\bar{\delta}}_{curr}$ from neighboring data points $\left(\dot{\delta}_1, \rho_1\right)$ and $\left(\dot{\delta}_2, \rho_2\right)$ is

$$\rho = \rho_1 + \frac{\ln\left(\frac{\dot{\bar{\delta}}_{curr}}{\dot{\delta}_1}\right)}{\ln\left(\frac{\dot{\delta}_2}{\dot{\delta}_1}\right)}\left(\rho_2 - \rho_1\right).$$

The formula for linear interpolation is

$$\rho = \rho_1 + \frac{\left(\dot{\bar{\delta}}_{curr} - \dot{\delta}_1\right)}{\left(\dot{\delta}_2 - \dot{\delta}_1\right)}\left(\rho_2 - \rho_1\right).$$

Tabular data as a function of slip/separation rate is often provided at evenly spaced intervals on a logarithmic scale. Consider logarithmic and linear interpolation schemes for $\dot{\bar{\delta}}_{curr} = 3$ between data points at $\dot{\delta}_1 = 1$ and $\dot{\delta}_2 = 10.$. Logarithmic interpolation gives ρ approximately halfway between ρ_1 and ρ_2; whereas linear interpolation gives ρ approximately 22% of the way between ρ_1 and ρ_2.

Abaqus/Explicit internally converts (or "regularizes") tabular dependencies to equally spaced intervals of independent variables to efficiently determine which data points to interpolate between. With logarithmic interpolation, these intervals are equally spaced on a logarithmic scale, often resulting in dramatically fewer intervals when data you provide is orders of magnitude over the slip/separation rates. Abaqus/Standard does not regularize tabular data.

Input File Usage: Use the following option to specify logarithmic interpolation (default):

**SURFACE INTERACTION*, RATE INTERPOLATION=LOGARITHMIC

Use the following option to specify linear interpolation:

**SURFACE INTERACTION*, RATE INTERPOLATION=LINEAR

Abaqus/CAE Usage: Specifying logarithmic or linear interpolation is not supported in Abaqus/CAE.

Contact Pressure-Overclosure Relationships

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Mechanical Contact Properties*
- *Output to the Output Database*
- **CONTACT CONTROLS*
- **SURFACE BEHAVIOR*
- *Creating interaction properties*
- *Customizing contact controls*

Overview

In Abaqus the following contact pressure-overclosure relationships can be used to define the contact model:

- the "hard" contact relationship minimizes the penetration of the secondary surface into the main surface at the constraint locations and does not allow the transfer of tensile stress across the interface;
- a "softened" contact relationship in which the contact pressure is a linear function of the clearance between the surfaces;
- a "softened" contact relationship in which the contact pressure is an exponential function of the clearance between the surfaces;
- a "softened" contact relationship in which a tabular pressure-overclosure curve is constructed by progressively scaling the default penalty stiffness (available only for general contact in Abaqus/Explicit);
- a "softened" contact relationship in which the contact pressure is a piecewise linear (tabular) function of the clearance between the surfaces; and
- a relationship in which there is no separation of the surfaces once they contact.

In addition, a viscous damping relationship can be defined that will affect the pressure-overclosure relationship; see *Contact Damping* for more information. In Abaqus/Standard pressure penetration loads can be applied to model fluid penetrating into the surface between two contacting bodies; see *Fluid Pressure Penetration Loads*.

Including a Contact Pressure-Overclosure Relationship in a Contact Property Definition

By default, a "hard" contact pressure-overclosure relationship is used for both surface-based contact and element-based contact. You can include a nondefault contact pressure-overclosure relationship in a specific contact property definition.

Input File Usage: Use both of the following options for surface-based contact:

**SURFACE INTERACTION*, NAME=*interaction_property_name*
**SURFACE BEHAVIOR*

Use both of the following options for element-based contact in Abaqus/Standard:

**INTERFACE* or **GAP*, ELSET=*name*
**SURFACE BEHAVIOR*

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method: Default**

Element-based contact is not supported in Abaqus/CAE.

Using the “Hard” Contact Relationship

The most common contact pressure-overclosure relationship is shown in *Figure 1*, although the zero-penetration condition might or might not be strictly enforced depending on the constraint enforcement method used (the constraint enforcement methods are discussed in *Contact Constraint Enforcement Methods in Abaqus/Standard* and *Contact Constraint Enforcement Methods in Abaqus/Explicit*). When surfaces are in contact, any contact pressure can be transmitted between them. The surfaces separate if the contact pressure reduces to zero. Separated surfaces come into contact when the clearance between them reduces to zero.

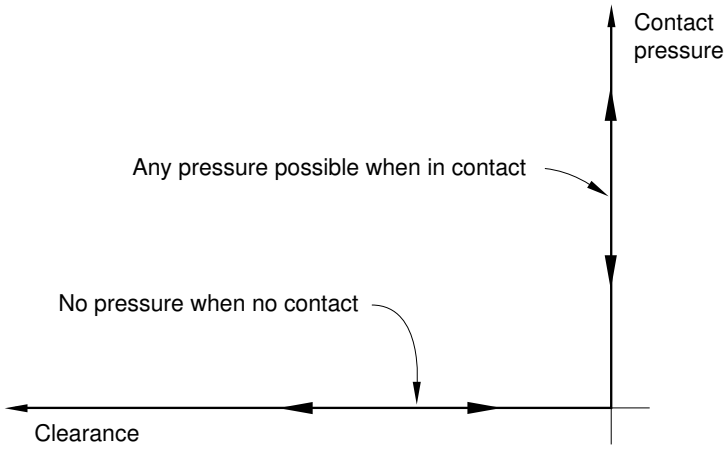


Figure 1: Default pressure-overclosure relationship.

Input File Usage: **SURFACE BEHAVIOR* (omit the PRESSURE-OVERCLOSURE parameter to obtain the default “hard” pressure-overclosure relationship)

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method: Default**: **Pressure-Overclosure: "Hard" Contact**

Considerations for Rigid-to-Rigid Contact in Abaqus/Explicit

For general contact in Abaqus/Explicit involving two rigid bodies with complete displacement control loading, the contact forces might be noisy due to effects of the number of contact interactions involving each rigid body and the contact penalty stiffness. This same effect will not occur with force control loading. The solution of the analysis should remain stable.

Using a “Softened” Contact Relationship

Three types of “softened” contact relationships are available in Abaqus. The pressure-overclosure relationship can be prescribed by using a linear law, a tabular piecewise-linear law, or an exponential law.

For contact involving element-based surfaces and for element-based contact (available only in Abaqus/Standard), the “softened” contact relationships are specified in terms of overclosure (or clearance) versus contact pressure. For contact involving a node-based surface or nodal contact elements (such as GAP and ITT elements) for which an area or length dimension is not defined, softened contact is specified in terms of overclosure (or clearance)

versus contact force. For secondary surfaces on beam-type elements in Abaqus/Standard and for the contact pair algorithm in Abaqus/Explicit, specify pressure as force per unit length. If the general contact algorithm in Abaqus/Explicit is being used for secondary surfaces on beam-type elements, specify pressure as force per unit area.

When using softened contact relationships that have nonzero pressure at zero overclosure in Abaqus/Explicit, you should be aware that initial, nonequilibrated contact pressures might be present in the analysis (see *Contact Initialization for Contact Pairs in Abaqus/Explicit*).

"Softened" Contact Versus "Hard" Contact

The "softened" contact pressure-overclosure relationships might be used to model a soft, thin layer on one or both surfaces. In Abaqus/Standard they are also sometimes useful for numerical reasons because they can make it easier to resolve the contact condition.

Using "Softened" Contact in Implicit Dynamic Simulations

Use the softened contact relationship with caution in implicit dynamic impact simulations. If this relationship is used in such a simulation, Abaqus/Standard will not use the impact algorithm, which destroys kinetic energy of the nodes on the surface when impact occurs, but will instead assume a perfectly elastic collision. The consequence of this change is that the secondary nodes bounce back immediately after impact with the main surface; hence, extensive "chattering" might result, leading to convergence problems and small time increments.

However, softened contact might work well in implicit dynamic calculations where impact effects are not important; for example, if contact changes are primarily due to sliding motion along a curved surface, such as might occur in low-speed metal forming applications.

Using "Softened" Contact in Explicit Dynamic Simulations

In Abaqus/Explicit softened contact can be enforced with either the kinematic or the penalty constraint enforcement method (see *Contact Constraint Enforcement Methods in Abaqus/Explicit* for details). With penalty enforcement the contact collisions are elastic except for the influence of contact damping, whereas with softened kinematic contact some energy will be absorbed by the impact because of algorithmic characteristics: the energy absorbed tends to increase as the contact stiffness increases. Another consideration is the effect on the time increment: with kinematic enforcement the stable time increment is independent of the contact stiffness, but with penalty contact the time increment decreases as the contact stiffness increases. Optionally, you can introduce contact mass scaling in general contact to avoid a reduction in the time increment (for more information, see *Mass Scaling to Account for Contact Stiffness*).

"Softened" Contact Defined as a Linear Function

In a linear pressure-overclosure relationship, the surfaces transmit contact pressure when the overclosure between them, measured in the contact (normal) direction, is greater than zero. The linear pressure-overclosure relationship is identical to a tabular relationship with two data points, where the first point is located at the origin.

You specify the slope of the pressure-overclosure relationship, k.

Input File Usage: **SURFACE BEHAVIOR*, PRESSURE-OVERCLOSURE=LINEAR
k

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method: Default**: **Pressure-Overclosure: Linear**, **Contact stiffness:** k

"Softened" Contact Defined in Tabular Form

To define a piecewise-linear pressure-overclosure relationship in tabular form, as shown in *Figure 2*, you specify data pairs (p_i, h_i) of pressure versus overclosure (where overclosure corresponds to negative clearance). You must specify the data as an increasing function of pressure and overclosure. In this relationship, the surfaces transmit contact pressure when the overclosure between them, measured in the contact (normal) direction, is greater than h_1, where h_1 is the overclosure at zero pressure. For overclosures greater than h_n, the pressure-overclosure relationship is extrapolated based on the last slope computed from the user-specified data (see *Figure 2*).

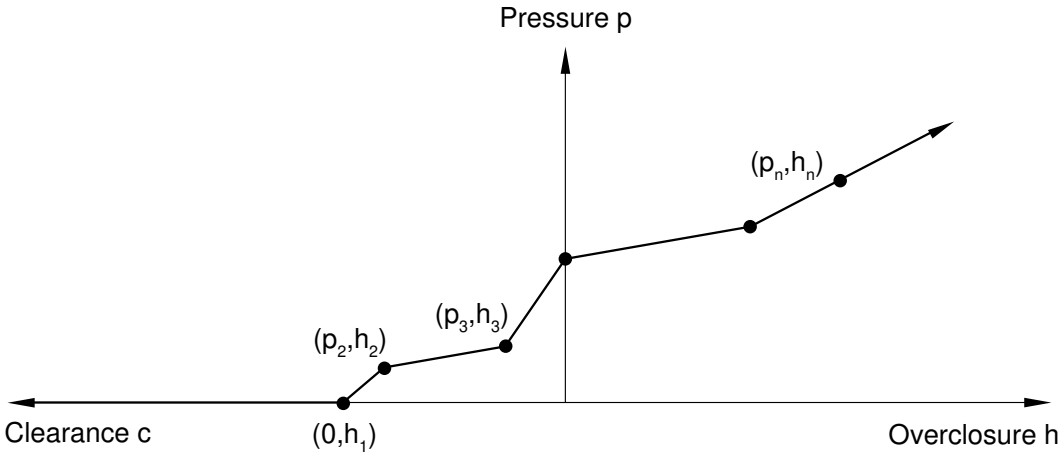


Figure 2: "Softened" pressure-overclosure relationship defined in tabular form.

Input File Usage: **SURFACE BEHAVIOR*, PRESSURE-OVERCLOSURE=TABULAR

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method: Default**: **Pressure-Overclosure: Tabular**

"Softened" Contact Defined as a Geometric Scaling of the Default Contact Stiffness

An alternative piecewise linear tabular pressure-overclosure relationship can be constructed by geometrically scaling the default contact stiffness. This model provides a simple interface to increase the default contact stiffness when a critical penetration is exceeded. A penetration measure, d, is defined either directly or as a fraction, r, of the minimum element length, L_{elem}, in the contact region. Each time the current penetration exceeds a multiple of this penetration measure, the contact stiffness is scaled by a factor, s (see *Figure 3*). The initial stiffness is set equal to the default contact stiffness, k_{dflt}, multiplied by a factor, s_0.

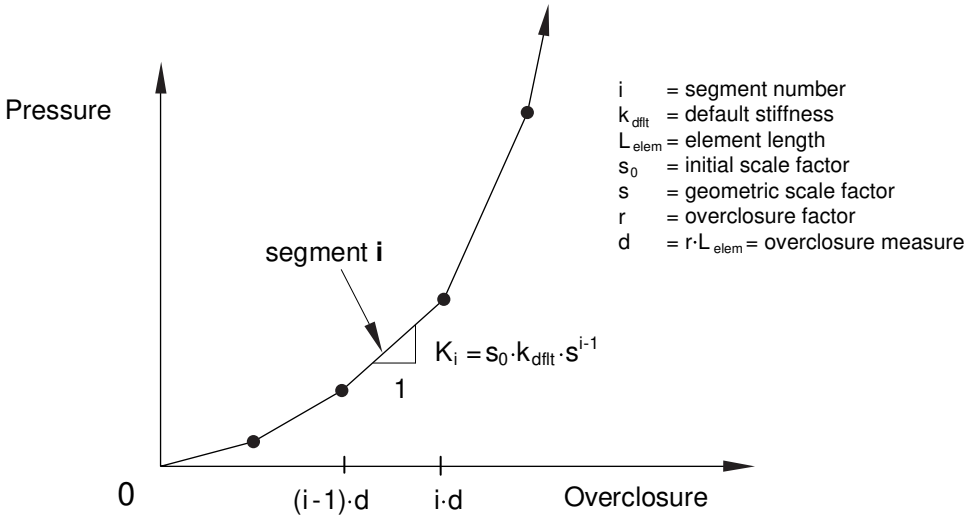


Figure 3: "Softened" scale factor pressure-overclosure relationship.

This option is available only for the general contact algorithm in Abaqus/Explicit.

Input File Usage: **SURFACE BEHAVIOR*, PRESSURE-OVERCLOSURE=SCALE FACTOR

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method: Default**: **Pressure-Overclosure: Scale Factor (General Contact)**

"Softened" Contact Defined with an Exponential Law

In an exponential (soft) contact pressure-overclosure relationship the surfaces begin to transmit contact pressure once the clearance between them, measured in the contact (normal) direction, reduces to c_0. The contact pressure transmitted between the surfaces then increases exponentially as the clearance continues to diminish. *Figure 4* illustrates this behavior in Abaqus/Standard.

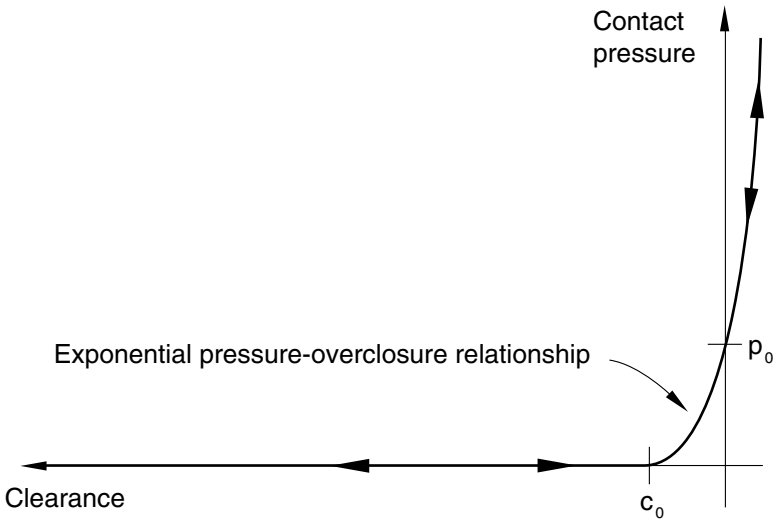


Figure 4: Exponential "softened" pressure-overclosure relationship in Abaqus/Standard.

In Abaqus/Explicit you can specify an optional limit on the contact stiffness that the model can attain, k_{max} (see *Figure 5*); this limit is useful for penalty contact to mitigate the effect that large stiffnesses have on reducing the stable time increment.

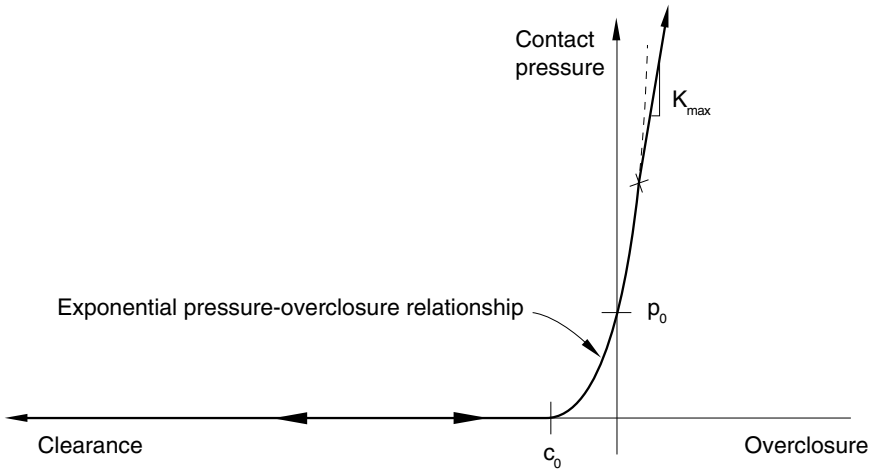


Figure 5: Exponential "softened" pressure-overclosure relationship in Abaqus/Explicit.

By default, k_{max} will be set to infinity for kinematic contact and to the default penalty stiffness for penalty contact.

You specify c_0; the contact pressure at zero clearance, p_0; and, optionally in Abaqus/Explicit, k_{max}.

Input File Usage: **SURFACE BEHAVIOR*, PRESSURE-OVERCLOSURE=EXPONENTIAL
c_0, p_0, k_{max}

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method: Default**: **Pressure-Overclosure: Exponential**, **Pressure p_0, Clearance c_0, Specify: k_{max}**

Using the No Separation Relationship

You can indicate that Abaqus should use the contact pressure-overclosure relationship that prevents surfaces from separating once they have come into contact.

The no separation relationship is often used with the rough friction model (see *Frictional Behavior*) to model nonintermittent, rough frictional contact. Using this combination of surface interaction models causes surfaces to remain fully bonded together (no separation and no tangential sliding) once they contact, even if the contact pressure between them is tensile. In Abaqus/Explicit the no separation relationship with the rough friction model cannot be used with adaptive meshing.

Input File Usage: **SURFACE BEHAVIOR*, NO SEPARATION

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method: Default**: **Pressure-Overclosure: Hard**, toggle off **Allow separation after contact**

Considerations for No Separation Relationship in Abaqus/Explicit with General Contact

With general contact in Abaqus/Explicit, no-separation behavior resists separation in the normal direction but typically does not prevent loss of contact as a result of one surface sliding off of another surface as shown in *Figure 6*.

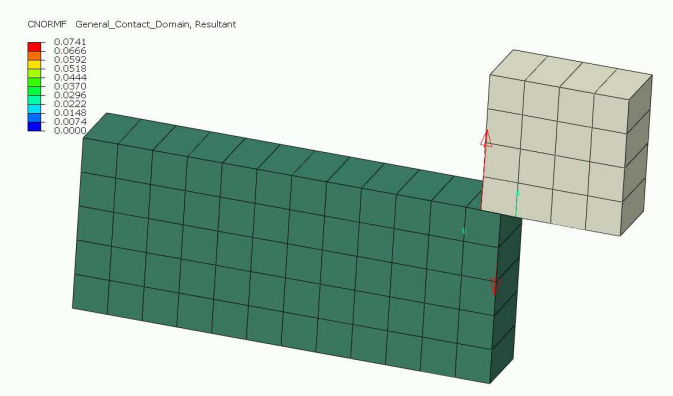


Figure 6: Sliding off mechanism.

The primary default behavior is desired in contexts such as hydrodynamic lubrication (for example, mimicking the pericardial cavity in heart models). However, this behavior is undesirable in other contexts, such as when modeling surface tension at a fluid-structure interface.

Instead of the fluid sliding off of the edge of a structure, surface tension should restrict sliding until the fluid bulges around a surface corner enough that conditions are met to continue sliding. To enable this specialized behavior, a no-separation behavior is in effect automatically for interactions involving surface tension properties. *Figure 7* shows an example of the fluid bulging at corners of a structure. The bulging is not large enough to reach the criterion for subsequent sliding in this example.

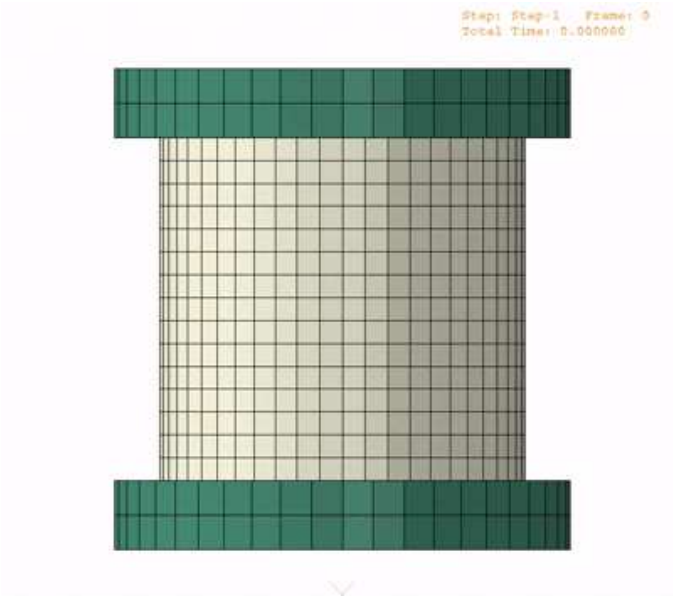


Figure 7: Surface tension effects around corners and edges.

"Softened" Contact with the No Separation Relationship in Abaqus/Explicit

If a softened contact relationship is specified with the no separation relationship in Abaqus/Explicit, the pressure-overclosure relationship includes tensile behavior. You cannot use the exponential and scale factor relationships with no separation behavior. For the tabular relationship, you must specify a point on the zero pressure axis; the slope continues into the tensile regime following the same slope as the first two data points (see *Figure 8*). The linear relationship will have a linear tensile pressure-overclosure relationship with the same slope that is used for the compressive behavior.

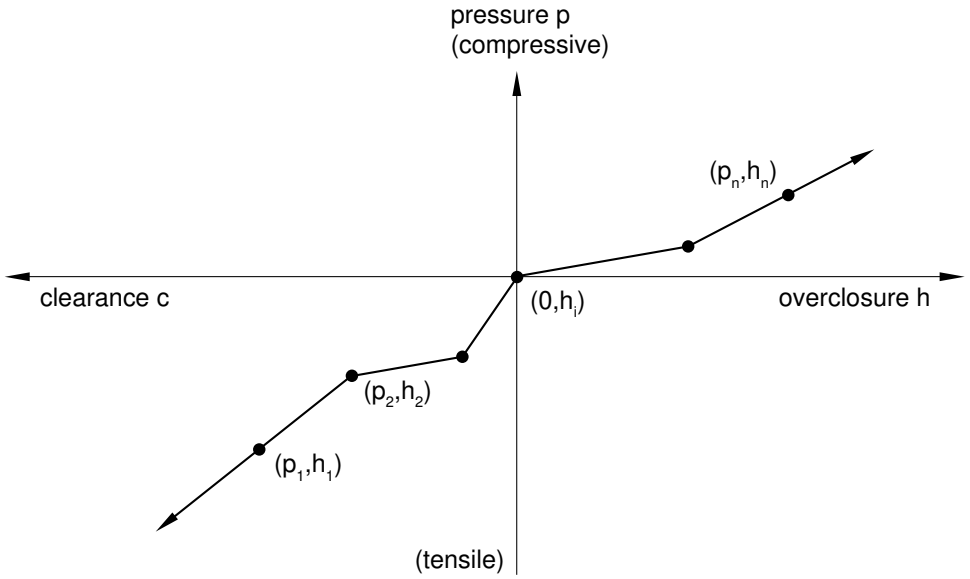


Figure 8: Piecewise linear "softened" pressure-overclosure relationship with tensile behavior in Abaqus/Explicit.

Surface Interaction Output Variables Related to the Contact Pressure-Overclosure

Abaqus/Standard provides both the clearance, COPEN, and the contact pressure, CPRESS, as output to the data, results, and output database files. Output to these files is requested as described in *Output to the Data and Results Files* and *Output to the Output Database*.

Abaqus/Explicit provides the contact pressure, CPRESS, as output to the output database file (see *Output to the Output Database* for details).

In the data, results, and output database files the output variable CPRESS gives the viscous damping pressures for an open secondary node. This variable also gives the contact pressure for a closed secondary node. In printed output a "VD" status indicates that the forces are for viscous damping.

Contours of the contact pressure on the secondary surface can be plotted in Abaqus/CAE.

Contact Damping

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Mechanical Contact Properties*
- **CONTACT DAMPING*
- *Creating interaction properties*

Overview

Contact damping:

- can be defined to oppose the relative motion between the interacting surfaces (in addition to the contact pressure-overclosure relationships discussed in *Contact Pressure-Overclosure Relationships* and the friction models discussed in *Frictional Behavior*);
- can affect both the motion normal and tangential to the surfaces;
- in the normal direction is proportional to the relative velocity between the surfaces;
- in the tangential direction is proportional to the relative tangential velocity in Abaqus/Standard and to the "elastic slip rate" associated with friction (see *Frictional Behavior* for a discussion of elastic slip) in Abaqus/Explicit—hence, in Abaqus/Explicit it does not resist the bulk of tangential sliding;
- is not applicable for linear perturbation procedures;
- in Abaqus/Standard it contributes to the force and stiffness definition and should generally be used only when it is otherwise impossible to obtain a solution—the best method for allowing a viscous pressure and shear stress to be transmitted between the contact surfaces in Abaqus/Standard to reduce convergence difficulties due to the sudden violation of contact constraints (common in some snap-through and buckling problems involving contact) is to specify the damping on a step-by-step basis using contact controls, as discussed in *Automatic Stabilization of Rigid Body Motions in Contact Problems*; and
- can be useful in Abaqus/Explicit to reduce solution noise—a small amount of viscous contact damping is used by default for softened contact and penalty contact in Abaqus/Explicit, as discussed below.

Defining Viscous Contact Damping for Relative Motions of Surfaces

In Abaqus/Standard the damping coefficient, μ, is a function of surface clearance, as shown in *Figure 1*. The damping coefficient is defined as a proportionality constant with units of pressure divided by velocity.

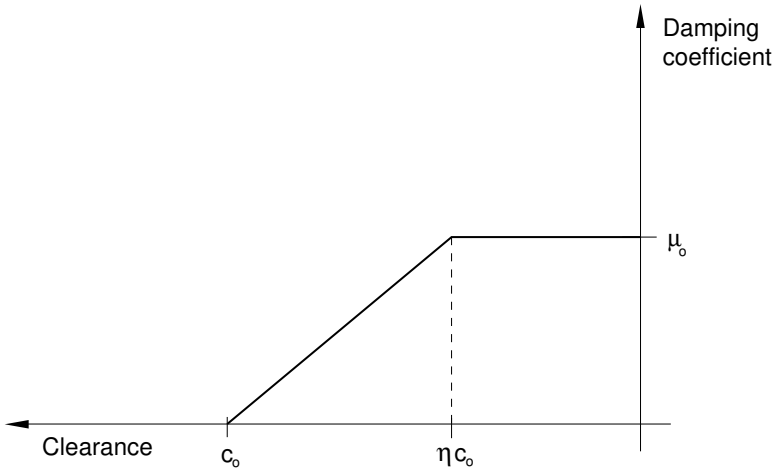


Figure 1: Damping coefficient-clearance relationship for viscous damping in Abaqus/Standard.

In Abaqus/Explicit the damping coefficient will remain at the specified constant value while the surfaces are in contact and at zero otherwise. The damping coefficient can be defined as a proportionality constant with units of pressure divided by velocity or as a unitless fraction of critical damping.

To define viscous damping, you must include it in a contact property definition.

Input File Usage: Use both of the following options for surface-based contact:

**SURFACE INTERACTION*, NAME=*interaction_property_name*
**CONTACT DAMPING*

Use both of the following options for element-based contact in Abaqus/Standard:

**INTERFACE* or **GAP*, ELSET=*name*
**CONTACT DAMPING*

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damping**

Element-based contact is not supported in Abaqus/CAE.

Damping and Pressure-Overclosure Relationships

In Abaqus/Standard the viscous damping relationship can be used with any contact relationship (see *Contact Pressure-Overclosure Relationships*).

In Abaqus/Explicit contact damping is not available for hard kinematic contact. Softened kinematic contact and all penalty contact will have default damping in the form of a critical damping fraction with $\mu_0 = 0.03$.

Specifying the Damping Coefficient Such That the Damping Force Is Directly Proportional to the Rate of Relative Motion between the Surfaces

You can specify damping directly in terms of the damping coefficient with units of pressure per velocity such that the damping forces will be calculated with $f_{vd} = \mu_0 A v_{rel}^{el}$, where A is the nodal area and v_{rel}^{el} is the rate of relative motion between the two surfaces.

For contact involving element-based surfaces and for element-based contact (available only in Abaqus/Standard), the damping coefficient is specified in terms of contact pressure. For contact involving a node-based surface or

nodal contact elements (such as GAP elements and ITT elements) for which an area or length dimension has not been defined, μ_0 must be specified as force per velocity. For secondary surfaces on beam-type elements, specify μ_0 as force per unit length per velocity.

Input File Usage: Use the following syntax in Abaqus/Standard:

**CONTACT DAMPING*, DEFINITION=DAMPING COEFFICIENT
μ_0, c_0, η

Use the following syntax in Abaqus/Explicit:

**CONTACT DAMPING*, DEFINITION=DAMPING COEFFICIENT
μ_0

Abaqus/CAE Usage: Use the following syntax in Abaqus/Standard:

Interaction module: contact property editor: **Mechanical->Damping**: **Definition: Damping coefficient**, **Linear** or **Bilinear**, **Damping Coeff.** μ_0, **Clearance** *c* and c_0 (η=0 for **Linear** and $\eta = \frac{c}{c_0}$ for **Bilinear**)

Use the following syntax in Abaqus/Explicit:

Interaction module: contact property editor: **Mechanical->Damping**: **Definition: Damping coefficient**, **Step**, **Damping Coeff.** μ_0

Specifying the Damping Coefficient as a Fraction of Critical Damping in Abaqus/Explicit

In Abaqus/Explicit you can specify a unitless damping coefficient in terms of the fraction of critical damping associated with the contact stiffness; this method is not available in Abaqus/Standard. The damping forces will be calculated with $f_{vd} = \mu_0 \sqrt{4mk_c} v_{rel}^{el}$, where *m* is the nodal mass, k_c is the nodal contact stiffness (in units of FL^{-1}), and v_{rel}^{el} is the rate of relative motion between the two surfaces.

Input File Usage: **CONTACT DAMPING*, DEFINITION=CRITICAL DAMPING FRACTION
critical damping fraction

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damping**: **Definition: Critical damping fraction**, **Crit. Damping Fraction** *critical damping fraction*

Specifying the Tangential Damping Coefficient

You can specify the ratio of the tangential damping coefficient to the normal damping coefficient, also called the tangent fraction.

The tangential damping uses the same form of damping as the normal damping. Tangential damping can be specified only in conjunction with normal damping. If tangential damping is activated in Abaqus/Standard, the damping stress is proportional to the relative tangential velocity. In Abaqus/Explicit tangential damping will be ignored if hard kinematic contact is used in the tangential direction or if friction is not defined. As stated previously, damping in the tangential direction in Abaqus/Explicit is proportional to the elastic slip rate (see *Frictional Behavior*) rather than the total rate of relative sliding.

For Abaqus/Standard the default value for the tangent fraction is 0.0; therefore, by default, the damping coefficient for the tangential direction is zero. For Abaqus/Explicit the default value for the tangent fraction is 1.0; therefore, by default, the damping coefficient for the tangential direction is equal to the damping coefficient for the normal direction. Furthermore, in Abaqus/Explicit softened contact and hard penalty contact have a default critical damping fraction of 0.03.

Input File Usage: **CONTACT DAMPING*, TANGENT FRACTION=*value*

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damping**: **Tangent fraction: Specify value:** *value*

Effect on the Increment Time in Abaqus/Explicit

Abaqus/Explicit accounts for the contact damping in the stable time increment calculation. Changing the default value might cause a significant drop in the stable time increment and, therefore, an increase in the solution time. Optionally, you can introduce contact mass scaling to the surface nodes in contact to avoid a reduction in the time increment (for more information, see *Mass Scaling to Account for Contact Stiffness*).

Choosing the Appropriate Coefficients for Viscous Damping in Abaqus/Standard

In Abaqus/Standard the appropriate magnitude for the local contact damping factor, μ_0, is problem-dependent. In some cases a simple calculation can be used to determine the magnitude; in other cases a reasonable value for μ_0 must be determined by trial and error. A reasonable value is one that has minimal impact on the solution prior to the unstable behavior in the model. A preliminary value can be found by looking at the contact pressures and velocities in the model before damping is added, as described below.

It may be difficult to determine the nodal velocities prior to the unstable behavior if output was not requested frequently. In such a situation the information in the message (`.msg`) file can be used to estimate the peak nodal velocity. By default, Abaqus/Standard provides the peak nodal displacement increment at every converged increment in this file. This displacement increment can be used along with the time increment to calculate a peak nodal velocity for the model. Although this velocity may not be very close to the actual relative velocity of the surfaces, it should be within an order of magnitude and is a reasonable value to use in calculating an initial viscous damping coefficient.

The maximum contact pressure between the surfaces also needs to be estimated. The viscous damping coefficient should then be set to a value that is a few orders of magnitude less than the ratio of the estimated maximum contact pressure over the calculated nodal velocity.

If it is not feasible to obtain the pressure and velocities as discussed above, a high damping value should be used initially and repeated analyses should be performed with smaller and smaller values. An appropriate value for μ_0 is one that is large enough to enable the analysis to get past any unstable response but not so large that the results at earlier or later times are affected significantly. *Snap-through buckling analysis of circular arches* demonstrates how the magnitude of the damping coefficient can be determined using the methods explained above.

The following example outlines how the value might be chosen for a typical case. Consider a simple modification to the two-dimensional Euler column buckling problem: add rigid surfaces parallel and on either side of the column so that the beam will contact the surfaces when it buckles. As the axial load is increased beyond the buckling load, the column will flatten out against the surface. Then, the midpoint of contact will lift off the surface and the beam will buckle into a higher mode. *Figure 2* shows this shape.

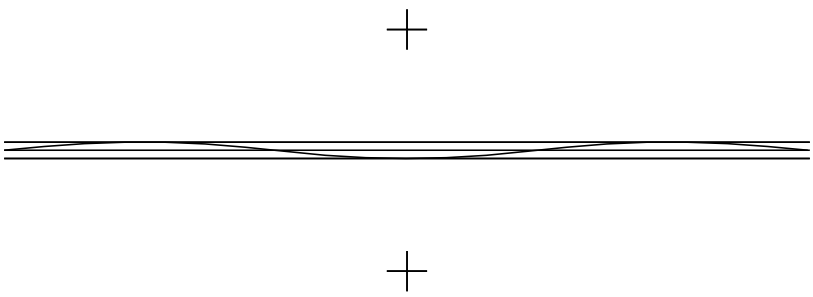


Figure 2: Constrained Euler buckling example for viscous damping.

When the column first buckles, the contact force, F, that the column exerts on one of the rigid surfaces can be approximated as

$$F \approx \frac{\pi^2 h}{4l} \left(P - P_{crit} \right),$$

where h is the separation distance between the rigid surfaces, l is the beam length, P is the applied load, and P_{crit} is the buckling load.

The approximation of the contact force entails the assumption that a single point comes into contact and that the shape of the buckled column does not change. The units of μ_0 are contact force per velocity, assuming that a node-based surface is used in this model. The velocity of the column, v, at the point of contact can be approximated as

$$v \approx \frac{h}{2\Delta t},$$

where Δt is the time increment. These estimates for the contact force and the column velocity give a value for the damping coefficient:

$$\mu_0 = \frac{F}{v} \approx \Delta t \frac{\pi^2}{2l} \left(P - P_{crit} \right).$$

This value can be used as a starting value, but different values should be tested.

Contact Blockage

Products: Abaqus/Explicit

References:

- *About Mechanical Contact Properties*
- *About Surface-Based Fluid Cavities*
- *Fluid Exchange Definition*
- **BLOCKAGE*
- **FLUID EXCHANGE ACTIVATION*
- **SURFACE INTERACTION*

Overview

The blockage of flow out of a cavity due to an obstruction caused by contacting surfaces:

- can be defined selectively for particular surfaces that may fully or partially cause the blockage; and
- can be accounted for only when the surfaces are used with the general contact algorithm.

Surfaces Used to Account for Contact Blockage

To consider an obstruction by contacting surfaces as discussed in *Accounting for Blockage due to Contacting Boundary Surfaces*, you must define a surface to represent the leakage area on the boundary of the fluid cavity. In addition, you must specify that the contacting surfaces can potentially cause blockage. All the surfaces (the surface on the boundary of the fluid cavity and the contacting surfaces) must be included in a general contact domain. To account for contact blockage, the nodes on the surfaces must be in node-to-face contact. When the nodes on the surface on the boundary of the fluid cavity come into contact with the contacting surfaces, the secondary nodes are marked as active nodes for contact blockage. The contact blockage is also considered in the edge-to-edge contact (see *Contact Formulation for General Contact in Abaqus/Explicit*).

Input File Usage: Use the following options to specify that two contacting surfaces can cause blockage:

**CONTACT PROPERTY ASSIGNMENT*
surface_1, surface_2, property_name
**SURFACE INTERACTION*, NAME=*property_name*
**BLOCKAGE*

Determining the Obstruction Area

Abaqus/Explicit determines the obstruction area by calculating the area fraction of the surface on the boundary of the fluid cavity that is not blocked by contacting surfaces. For each element face of this surface representing the leakage area, the blocked area is calculated based on the active nodes for contact blockage. The element blocked area is determined by

$$A_{eb} = A_e \frac{N_c}{N_e},$$

where A_{eb} is the element blocked area, A_e is the element area, N_e is the total number of element nodes, and N_c is the total number of active nodes for contact blockage in the element. The element is fully blocked by the contacting surfaces when all element nodes are active for contact blockage. The total obstruction area is the sum of all the element blocked areas. The leakage area used in the fluid exchange calculation is obtained by subtracting the total obstruction area from the total area of the surface if the effective area is not specified for the fluid exchange. If both the effective area and a surface are specified (see *Fluid Exchange Definition*), the leakage area used in the fluid exchange calculation is obtained by using the ratio of the total obstruction area to the total area of the surface multiplied by the effective area. In this case a node-based surface can be used, and the leakage area is obtained by using the ratio of the total active contact blockage nodes to the total number of nodes defined in the surface.

Frictional Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Mechanical Contact Properties*
- *FRIC*
- *FRIC_COEF*
- *VFRIC*
- *VFRIC_COEF*
- *VFRICTION*
- **FRICTION*
- **CHANGE FRICTION*
- **CONTACT CONTROLS ASSIGNMENT*
- *Creating interaction properties*

Overview

When surfaces are in contact they usually transmit shear as well as normal forces across their interface. There is generally a relationship between these two force components. The relationship, known as the friction between the contacting bodies, is usually expressed in terms of the stresses at the interface of the bodies. The friction models available in Abaqus:

- include the classical isotropic Coulomb friction model (see *Coulomb friction*), which in Abaqus:
  - allows the friction coefficient to be defined in terms of slip rate, contact pressure, average surface temperature at the contact point, and field variables; and
  - provides the option for you to define a static and a kinetic friction coefficient with a smooth transition zone defined by an exponential curve;
- allow the introduction of a shear stress limit, τ_{max}, which is the maximum value of shear stress that can be carried by the interface before the surfaces begin to slide;
- include anisotropic extensions of the basic Coulomb friction model;
- include an option in Abaqus/Explicit in which the nominal friction coefficient for a contact interaction is derived from coefficients specified as surface properties;
- include a model that eliminates frictional slip when surfaces are in contact;
- include a "softened" interface model for sticking friction in Abaqus/Explicit in which the shear stress is a function of elastic slip;
- can be enforced with a stiffness (penalty) method, a kinematic method (in Abaqus/Explicit), or a Lagrange multiplier method (in Abaqus/Standard), depending on the contact algorithm used; and
- can be defined in user subroutines *FRIC* or *FRIC_COEF* (in Abaqus/Standard) or *VFRIC*, *VFRICTION*, or *VFRIC_COEF* (in Abaqus/Explicit).

In Abaqus/Standard tangential damping forces can be introduced proportional to the relative tangential velocity, while in Abaqus/Explicit tangential damping forces can be introduced proportional to the rate of relative elastic slip between the contacting surfaces (see *Contact Damping* for more information).

Two methods exist to define the elastic slip for the stiffness (penalty) method and to define the Lagrange multiplier method. The recommended method is to define surface pairwise interactions through contact controls. Alternatively, you can define interactions through the friction property.

Including Friction Properties in a Contact Property Definition

Abaqus assumes by default that the interaction between contacting bodies is frictionless. You can include a friction model in a contact property definition for both surface-based contact and element-based contact.

Input File Usage: Use both of the following options for surface-based contact:

**SURFACE INTERACTION*, NAME=*interaction_property_name*
**FRICTION*

Use both of the following options for element-based contact in Abaqus/Standard:

**INTERFACE* or **GAP*, ELSET=*name*
**FRICTION*

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical**->**Tangential Behavior**

Element-based contact is not supported in Abaqus/CAE.

Changing Friction Properties during an Analysis

The methods used to change friction properties during an analysis differ between Abaqus/Standard and Abaqus/Explicit.

Changing Friction Properties during an Abaqus/Standard Analysis

It is possible to remove, to modify, or to add a friction model that does not involve a user subroutine to a contact property definition in any particular step of an Abaqus/Standard simulation. In some models, such as shrink-fit contact interference problems, friction should not be added until after the first steps have been completed. In other models friction might be removed or lowered to represent the introduction of a lubricant between the bodies.

You must identify which contact property definition or contact element set is being changed.

Input File Usage: Use both of the following options for surface-based contact:

**CHANGE FRICTION*, INTERACTION=*name*
**FRICTION*

Use both of the following options for element-based contact:

**CHANGE FRICTION*, ELSET=*name*
**FRICTION*

Abaqus/CAE Usage: Define a contact property with a new friction definition. Then change the contact property assigned to an interaction in a particular step.

Interaction module:

Contact property editor: **Mechanical**->**Tangential Behavior**

Interaction editor: **Contact interaction property:** *new_interaction_property_name*

Element-based contact is not supported in Abaqus/CAE.

Specifying the Time Variation of the Change in Friction Properties

You can specify an amplitude curve (see *Amplitude Curves*) to define the time variation of changes in friction coefficients and, if applicable, allowable elastic slip (see *Stiffness Method for Imposing Frictional Constraints in Abaqus/Standard* below) throughout the step. If you do not specify an amplitude curve, changes in these friction properties are either applied immediately at the beginning of the step or ramped up linearly over the step, depending on the default amplitude variation assigned to the step (see *Defining an Analysis*), with some exceptions as described below. For many step types the default transition type is a linear ramping from old to new values, which helps avoid convergence problems that can occur upon sudden changes in friction properties.

Amplitude curves used to control variations in friction properties are subjected to the following restrictions:

- a tabular or smooth step amplitude definition must be used,
- only amplitudes with monotonically increasing values between 0.0 and 1.0 are accepted, and
- the amplitude must be defined in terms of step time and using relative magnitudes.

The value of a friction coefficient or allowable elastic slip in effect at a given time is typically equal to the value of the property at the start of the step plus the current amplitude value times the anticipated change in property value over the step. Variations in friction properties must consider the following:

- Changes in the type of frictional constraint enforcement method (penalty or Lagrange multiplier methods), changes between a “rough” friction model and a finite friction coefficient, and changes to friction properties other than the friction coefficient or allowable elastic slip always occur at the beginning of a step.
- If a friction coefficient is dependent on slip rate, contact pressure, average surface temperature at the contact point, or field variables, the estimate of the final value of the friction coefficient for the step (which is used in calculating the anticipated change in the friction coefficient over the step) assumes that the current slip rate, contact pressure, etc. will remain in effect at the end of the step.
- If a friction coefficient is changed during the first step of an analysis, its value at the start of the step is equal to zero for this calculation, regardless of the original friction definition in the model.
- Changes in allowable elastic slip always occur at the beginning of a step when an exponential-decay friction model is used or when frictional properties are changed during the first general step or during a steady-state transport step that is preceded by a step type other than steady-state transport.

Input File Usage: **CHANGE FRICTION*, AMPLITUDE=*name*

Abaqus/CAE Usage: Time-dependent changes in friction properties are not supported in Abaqus/CAE.

Resetting the Frictional Properties to Their Default Values

You can reset the frictional properties of the specified contact property definition or element set to their original values.

Input File Usage: Use either of the following options:

**CHANGE FRICTION*, RESET, INTERACTION=*name*
**CHANGE FRICTION*, RESET, ELSET=*name*

In this case the **FRICTION* option is not needed.

Abaqus/CAE Usage: Interaction module:

Contact property editor: **Mechanical->Tangential Behavior**: **Friction formulation: Frictionless**

Interaction editor: **Contact interaction property:** *default_interaction_property_name*

Changing Friction Properties during an Abaqus/Explicit Analysis

In Abaqus/Explicit the friction definition is specified as part of the model definition for a general contact analysis and as part of the history definition for a contact pair analysis. See *Assigning Contact Properties for General Contact in Abaqus/Explicit* and *Assigning Contact Properties for Contact Pairs in Abaqus/Explicit* for information on changing aspects of any contact property definition during an Abaqus/Explicit analysis.

Using the Basic Coulomb Friction Model

The basic concept of the Coulomb friction model is to relate the maximum allowable frictional (shear) stress across an interface to the contact pressure between the contacting bodies. In the basic form of the Coulomb friction model, two contacting surfaces can carry shear stresses up to a certain magnitude across their interface before they start sliding relative to one another; this state is known as sticking. The Coulomb friction model defines this critical shear stress, τ_{crit}, at which sliding of the surfaces starts as a fraction of the contact pressure, p, between the surfaces ($\tau_{crit} = \mu p$). The stick/slip calculations determine when a point transitions from sticking to slipping or from slipping to sticking. The fraction, μ, is known as the coefficient of friction.

For the case when the secondary surface consists of a node-based surface, the contact pressure is equal to the normal contact force divided by the cross-sectional area at the contact node. In Abaqus/Standard the default cross-sectional area is 1.0; you can specify a cross-sectional area associated with every node in the node-based surface when the surface is defined or, alternatively, assign the same area to every node through the contact property definition. In Abaqus/Explicit the cross-sectional area is always 1.0, and you cannot change it.

The basic friction model assumes that μ is the same in all directions (isotropic friction). For a three-dimensional simulation there are two orthogonal components of shear stress, τ_1 and τ_2, along the interface between the two bodies. These components act in the local tangent directions for the contact surfaces or contact elements. The local tangent directions for contact surfaces are defined in *Contact Formulations in Abaqus/Standard*, and those for contact elements are defined in the sections describing contact modeling with those elements.

Abaqus combines the two shear stress components into an "equivalent shear stress," τ_{eq}, for the stick/slip calculations, where $\tau_{eq} = \sqrt{\tau_1^2 + \tau_2^2}$. In addition, Abaqus combines the two slip velocity components into an equivalent slip rate, $\dot{\gamma}_{eq} = \sqrt{\dot{\gamma}_1^2 + \dot{\gamma}_2^2}$. Abaqus/Explicit general contact uses a filtered slip rate measure as described in *About Mechanical Contact Properties*. The stick/slip calculations define a surface (see *Figure 1* for a two-dimensional representation) in the contact pressure–shear stress space along which a point transitions from sticking to slipping.

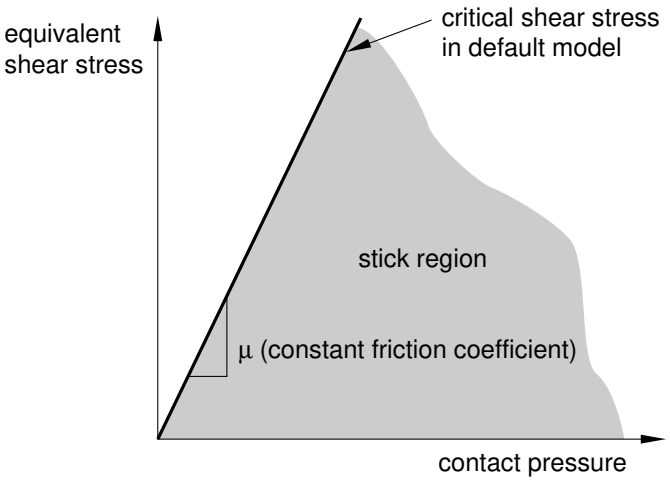


Figure 1: Slip regions for the basic Coulomb friction model.

There are two ways to define the basic Coulomb friction model in Abaqus. In the default model the friction coefficient is defined as a function of the equivalent slip rate and contact pressure. Alternatively, you can specify the static and kinetic friction coefficients directly.

Specifying a Constant or Tabular-Dependent Coefficient of Friction

In general, you can specify tabular dependence of a coefficient of friction as

$$\mu = \mu\left(\dot{\gamma}_{eq}, p, \overline{\theta}, \overline{f}^{\alpha}\right),$$

where $\dot{\gamma}_{eq}$ is the equivalent slip rate, p is the contact pressure, $\overline{\theta} = \frac{1}{2}\left(\theta_A + \theta_B\right)$ is the average temperature at the contact point, and $\overline{f}^{\alpha} = \frac{1}{2}\left(f_A^{\alpha} + f_B^{\alpha}\right)$ is the average predefined field variable α at the contact point. θ_A, θ_B, f_A^{α}, and f_B^{α} are the temperature and predefined field variables at points A and B on the surfaces. Point A is a node on the secondary surface, and point B corresponds to the nearest point on the opposing main surface. The temperature and field variables are interpolated along the surface at location B. If the main surface consists of a rigid body, the temperature and field variable at the reference node are used.

The friction coefficient can depend on slip rate, contact pressure, temperature, and field variables. By default, it is assumed that the friction coefficients do not depend on field variables. In the simple case of a constant friction coefficient, a single coefficient is specified with no dependence.

The coefficient of friction can be set to any nonnegative value. A zero friction coefficient means that no shear forces will develop and the contact surfaces are free to slide. You do not need to define a friction model for such a case.

For a discussion of interpolating tabular dependence with respect to rates of relative motion, see *About Mechanical Contact Properties*.

Input File Usage: **FRICTION*, DEPENDENCIES=*n*

μ, $\dot{\gamma}_{eq}$, p, $\overline{\theta}$, $\overline{f}^{\alpha}$

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical**->**Tangential Behavior**: **Friction formulation: Penalty**: **Friction**

If necessary, toggle on **Use slip-rate-dependent data**, **Use contact-pressure-dependent data**, and/or **Use temperature-dependent data**; and/or

specify the **Number of field variable** dependencies in addition to slip rate, contact pressure, and temperature.

Specifying Static and Kinetic Friction Coefficients

Experimental data show that the friction coefficient that opposes the initiation of slipping from a sticking condition is different from the friction coefficient that opposes established slipping. The former is typically referred to as the "static" friction coefficient, and the latter is referred to as the "kinetic" friction coefficient. Typically, the static friction coefficient is higher than the kinetic friction coefficient.

In the default model the static friction coefficient corresponds to the value given at zero slip rate, and the kinetic friction coefficient corresponds to the value given at the highest slip rate. The transition between static and kinetic friction is defined by the values given at intermediate slip rates. In this model the static and kinetic friction coefficients can be functions of contact pressure, temperature, and field variables.

Abaqus also provides a model to specify a static and a kinetic friction coefficient directly. In this model it is assumed that the friction coefficient decays exponentially from the static value to the kinetic value according to the formula:

$$\mu = \mu_k + (\mu_s - \mu_k)\, e^{-d_c \dot{\gamma}_{eq}},$$

where μ_k is the kinetic friction coefficient, μ_s is the static friction coefficient, d_c is a user-defined decay coefficient, and $\dot{\gamma}_{eq}$ is the slip rate (see Oden, J. T. and J. A. C. Martins, 1985). This model can be used only with isotropic friction and does not allow dependence on contact pressure, temperature, or field variables. There are two ways of defining this model.

Providing the Static, Kinetic, and Decay Coefficients Directly

You can provide the static friction coefficient, the kinetic friction coefficient, and the decay coefficient directly (see *Figure 2*).

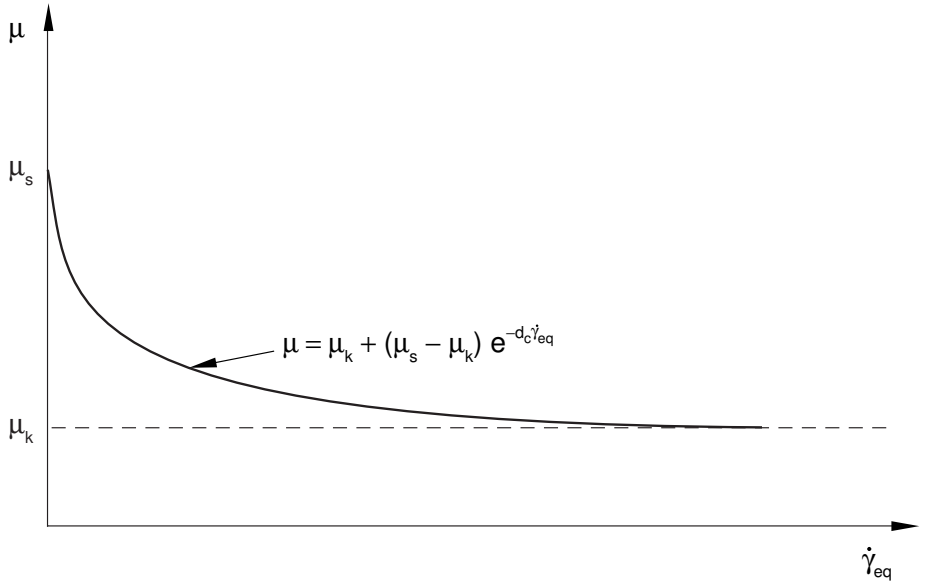


Figure 2: Exponential decay friction model.

Input File Usage: **FRICTION*, EXPONENTIAL DECAY
μ_s, μ_k, d_c

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical**->**Tangential Behavior**: **Friction formulation: Static-Kinetic Exponential Decay**: **Friction**, **Definition: Coefficients**

Using Test Data to Fit the Exponential Model

Alternatively, you can provide test data points to fit the exponential model. At least two data points must be provided. The first point represents the static coefficient of friction specified at $\dot{\gamma}_{eq} = 0.0$, and the second point, ($\dot{\gamma}_2$, μ_2) (shown in *Figure 3*), corresponds to an experimental measurement taken at a reference slip rate $\dot{\gamma}_2$. An additional data point can be specified to characterize the exponential decay. If this additional data point is omitted, Abaqus will automatically provide a third data point, ($\dot{\gamma}_\infty$, μ_∞), to model the assumed asymptotic value of the friction coefficient at infinite velocity. In such a case μ_∞ is chosen such that $(\mu_2 - \mu_\infty) / (\mu_1 - \mu_\infty) = 0.05$.

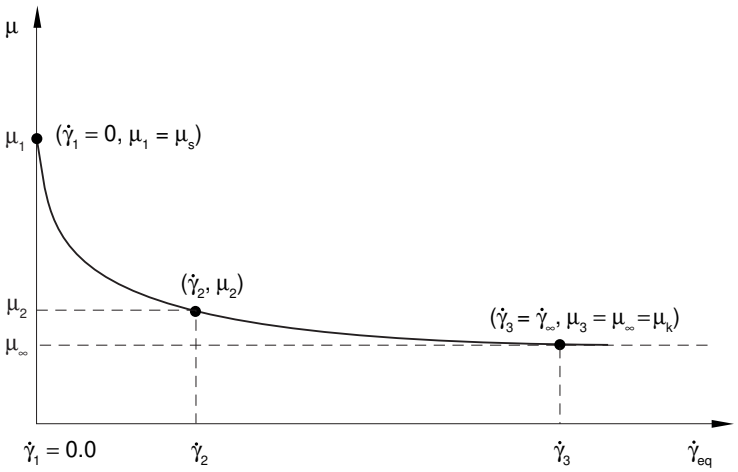


Figure 3: Exponential decay friction model specified with test data points.

Input File Usage: **FRICTION*, EXPONENTIAL DECAY, TEST DATA

μ_1

μ_2, $\dot{\gamma}_2$

μ_∞

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical**->**Tangential Behavior**: **Friction formulation: Static-Kinetic Exponential Decay**: **Friction**, **Definition: Test data**

Deriving Friction Coefficients from Quantities Specified as Surface Properties

In Abaqus/Explicit you can establish friction coefficients as mathematical combinations of coefficients specified as surface properties. For example, you can assign a particular friction coefficient to a surface associated with all steel parts, you can assign a second friction coefficient to a surface associated with all rubber parts, and likewise for other materials. These surface-based friction coefficients apply to interactions between the same material. A combinatorial rule is used to determine friction coefficients for interactions between different materials. You can override approximate friction coefficients computed in this manner by the traditional approach of assigning friction coefficients as contact property assignments based on combinations of surfaces.

The combinatorial approach reduces the required user input. For example, a simulation involving six materials could involve contact interactions with 21 unique material combinations, as shown in *Figure 4*. For simulations involving many materials, it might suffice to

- determine approximate friction coefficients for contact between like surfaces (corresponding to entries 1 through 6 of the table of *Figure 4*) and a subset of other surface combinations from experiments or available references; and
- allow a combinatorial rule to determine friction coefficients for the remaining surface combinations, at least for early stages of a design.

		Material type on one side of contact interface					
		Mat 1	Mat 2	Mat 3	Mat 4	Mat 5	Mat 6
Material type on other side of contact interface	Mat 1	1	7	8	9	10	11
	Mat 2		2	12	13	14	15
	Mat 3			3	16	17	18
	Mat 4				4	19	20
	Mat 5					5	21
	Mat 6						6

Figure 4: Example with six materials and 21 unique combinations of two materials in contact.

Abaqus/Explicit uses the equation

$$\mu_{AB} = \left[1 + \alpha\left(1 - \frac{\mu_A}{\mu_B}\right)\right]\mu_A,$$

where $\mu_A \leq \mu_B$ to compute the friction coefficient, μ_{AB}, for an interaction between surfaces A and B if μ_A and μ_B are assigned as surface properties to the respective surfaces. With this combinatorial rule, μ_{AB} is at most $1 + \alpha$ times greater than the smaller of the two surface-based friction coefficients. The default value of α is 0.3.

Consider an example with $\mu_1 = 0.05$, $\mu_2 = 0.1$, and $\mu_3 = 0.3$. The default combinatorial rule gives $\mu_{12} = 0.0575$, $\mu_{13} = 0.0625$, and $\mu_{23} = 0.12$. For example, you may choose to override the value of μ_{23} using the traditional contact property assignment approach.

The user interface for assigning surface-based friction coefficients is discussed in *Surface-Based Friction Coefficients*.

Input File Usage: Use the following option to adjust the value of α:

**COMBINATORIAL RULE*, ALPHA=α

Use the following option to indicate that friction coefficients assigned as contact properties take precedence over friction coefficients derived from values specified as surface properties (default):

**FRICTION*, COEFFICIENT PRECEDENCE=CONTACT PROPERTY

Use the following option to indicate that friction coefficients derived from values specified as surface properties take precedence over friction coefficients assigned as contact properties:

*FRICTION, COEFFICIENT PRECEDENCE=SURFACE PROPERTY COMBINATION

Abaqus/CAE Usage: Deriving friction coefficients from quantities specified as surface properties is not supported in Abaqus/CAE.

Using the Optional Shear Stress Limit

You can specify an optional equivalent shear stress limit, τ_{max}, so that, regardless of the magnitude of the contact pressure stress, sliding will occur if the magnitude of the equivalent shear stress, τ_{eq}, reaches this value (see *Figure 5*). A value of zero is not allowed.

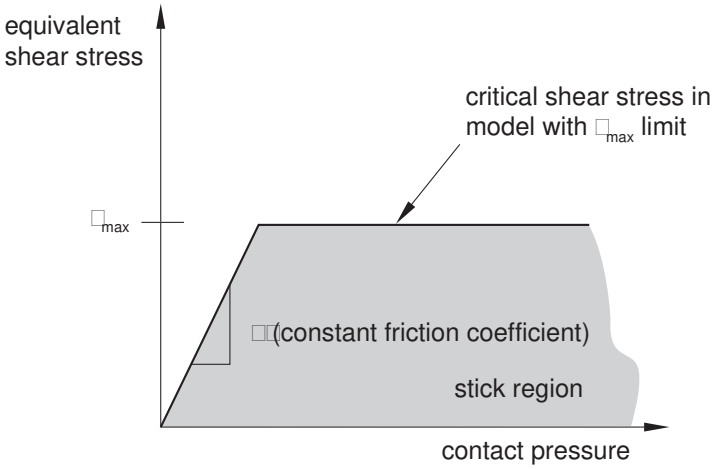


Figure 5: Slip regions for the friction model with a limit on the critical shear stress.

This shear stress limit is typically introduced in cases when the contact pressure stress might become very large (as can happen in some manufacturing processes), causing the Coulomb theory to provide a critical shear stress at the interface that exceeds the yield stress in the material beneath the contact surface. A reasonable upper bound estimate for τ_{max} is $\sigma_y/\sqrt{3}$, where σ_y is the Mises yield stress of the material adjacent to the surface; however, empirical data are the best source for τ_{max}.

Input File Usage: *FRICTION, TAUMAX=τ_{max}

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical**->**Tangential Behavior**: **Friction formulation: Penalty** or **Lagrange Multiplier**: **Shear Stress**, **Shear stress limit: Specify:** τ_{max}

Limitations with the Shear Stress Limit

In Abaqus/Explicit a shear stress limit cannot be used when a contact pair uses a node-based surface as one of the surfaces.

Anisotropic Friction with Directional Preference as a Surface Property

You can specify an anisotropic friction model in Abaqus/Explicit for which directional preferences are specified as surface properties, while the nominal, or average, friction coefficient is specified as a contact interaction

property in the same manner as for isotropic friction. The resulting critical contact shear stress surface is elliptical in the τ_1–τ_2 plane, as shown in *Figure 6*. Points on the critical shear stress surface satisfy the equation:

$$\sqrt{\tau_1^2 + \tau_2^2} = s_{combined}(\theta)\,\mu_{nom} P,$$

where $s_{combined}$ represents the combined effects of surface-based directional preferences (and these combined effects evolve as the relative surface orientations change), μ_{nom} is the specified nominal (average) friction coefficient, and P is the contact pressure. Maximum and minimum values of $s_{combined}$, corresponding to directions along the major and minor axes of the critical shear stress surface, are $s^{\max}_{combined} = 1 + \epsilon_{combined}$ and $s^{\min}_{combined} = 1 - \epsilon_{combined}$.

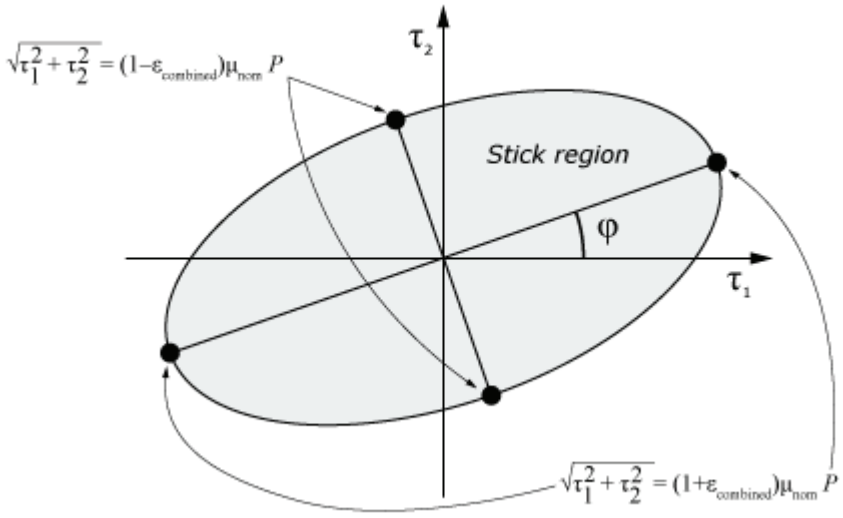


Figure 6: Critical shear stress surface for anisotropic friction.

ϵ is the frictional directional preference factor. It is a unitless parameter that can range from –1.0 to 1.0 and is a measure of the eccentricity of the scaling ellipse. The most commonly used eccentricity measure for ellipses is $e = \frac{c}{a}$ (see *Figure 7*). The relationship between ϵ and e is:

$$\left(\frac{b}{a}\right)^2 = \frac{(1-\epsilon)^2}{(1+\epsilon)^2},$$

$$e = 2 * \frac{\sqrt{\epsilon}}{(1+\epsilon)}.$$

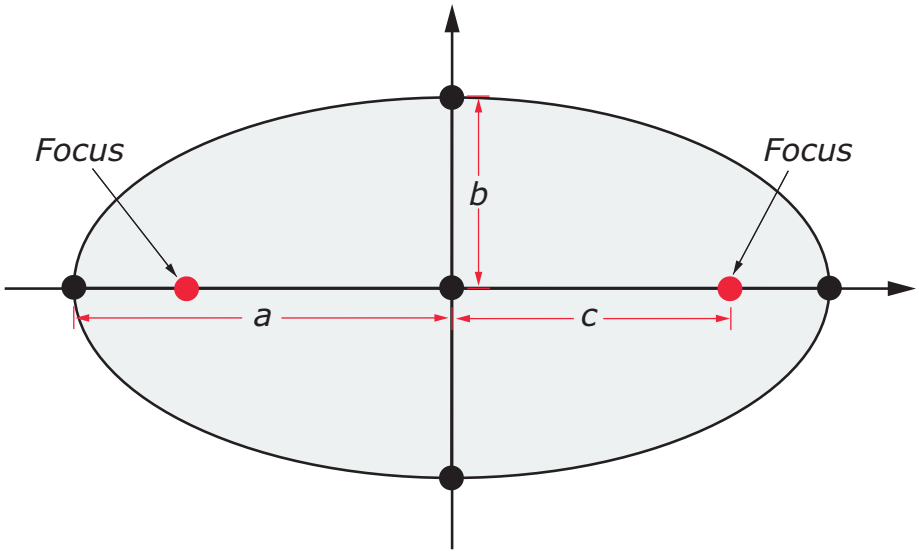


Figure 7: Geometrical quantities for an ellipse.

The critical contact shear stress surface influences the friction algorithm as follows:

1. Abaqus/Explicit computes a candidate contact shear stress necessary to enforce stick conditions:
$$\boldsymbol{\tau}^{cand} = \boldsymbol{\tau}^{stick} = \left(\tau_1^{stick}, \tau_2^{stick}\right).$$

2. If $\boldsymbol{\tau}^{cand}$ lies on or within the critical shear stress surface, as shown in *Figure 8*, Abaqus/Explicit accepts the candidate contact shear stress as the current contact shear stress, such that stick conditions are in effect.

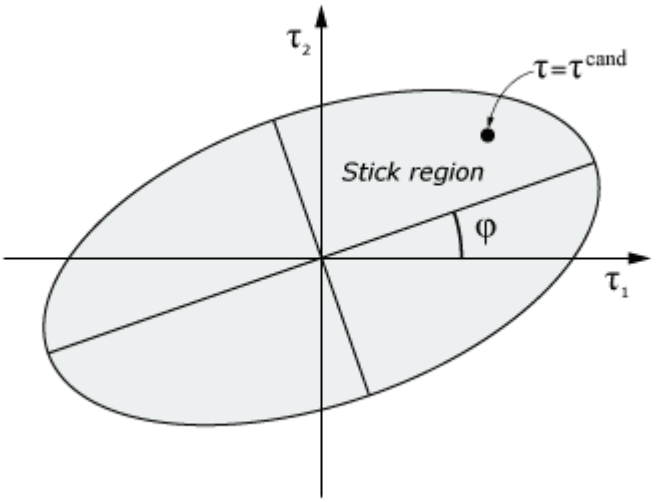


Figure 8: Example of candidate contact shear stress within the sticking region for anisotropic friction.

3. Otherwise, if $\boldsymbol{\tau}^{cand}$ lies outside the critical shear stress surface, Abaqus/Explicit sets the contact shear stress equal to $\boldsymbol{\tau}^{crit}$ on the critical shear stress surface where the normal to the critical shear stress surface passes through $\boldsymbol{\tau}^{cand}$, as shown in *Figure 9*, and sets the direction of incremental slip to be normal to the critical shear stress surface.

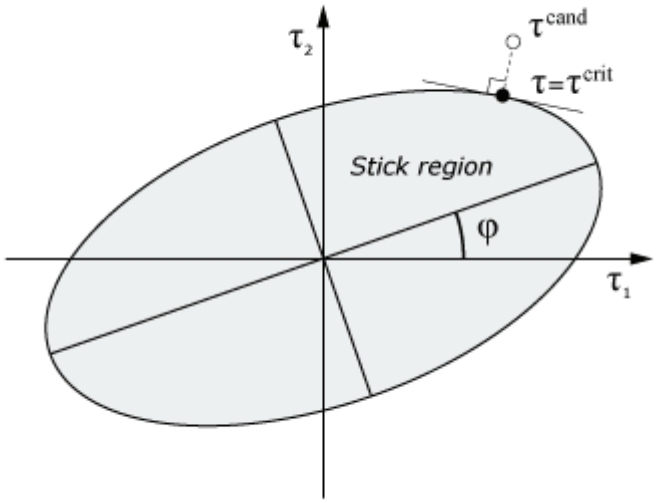


Figure 9: Example of candidate contact shear stress outside of the sticking region for anisotropic friction.

$s_{combined}$ can be thought of as a scaling ellipse calculated as a weighted average of surface-based scaling ellipses:

$$s_{combined}(\theta) = w_1 s_{surf1}(\theta) + w_2 s_{surf2}(\theta),$$

where the weight factors sum to unity ($w_1 + w_2 = 1$). s_{surf1} and s_{surf2} are scaling ellipses representing directional preferences of respective surfaces at a contact location. The maximum and minimum principal values of each scaling ellipse are of the form $s_i = 1 + \epsilon_i$ and $s_i = 1 - \epsilon_i$, respectively. A lack of directional preference corresponds to $\epsilon_i = 0$.

If both surfaces of a contact interaction contribute directional preference to the frictional behavior, the shape of the $s_{combined}$ scaling ellipse evolves as the (relative) orientations of the contacting surfaces change. For example, for contact between like surfaces with equal weighting factors ($w_1 = w_2 = \frac{1}{2}$):

$$\epsilon_{combined} = \epsilon_s \cos(\gamma),$$

where $\epsilon_s = \epsilon_{surf1} = \epsilon_{surf2}$ and γ is the angle between major axes of the surface scaling ellipses across the contact interface. Consider the following specific cases for contact between like surfaces:

- $\gamma = 0^\circ$: This corresponds to aligned directional preferences of contacting surfaces, as shown in Case 1 of *Figure 10* and *Figure 11*. In this situation $\epsilon_{combined} = \epsilon_s$ and $s_{combined} = s_{surf1} = s_{surf2}$.
- $\gamma = 90^\circ$: This corresponds to orthogonal directional preferences of contacting surfaces, as shown in Case 2 of *Figure 10* and *Figure 11*. In this situation opposing direction preferences of the surfaces cancel each other, such that $\epsilon_{combined} = 0$ and $s_{combined}$ corresponds to a unit circle.
- $\gamma = 45^\circ$: $\epsilon_{combined} = \frac{\sqrt{2}}{2}\epsilon_s$ and the major axis of $s_{combined}$ bisects the major axes of the respective surface scaling ellipses).

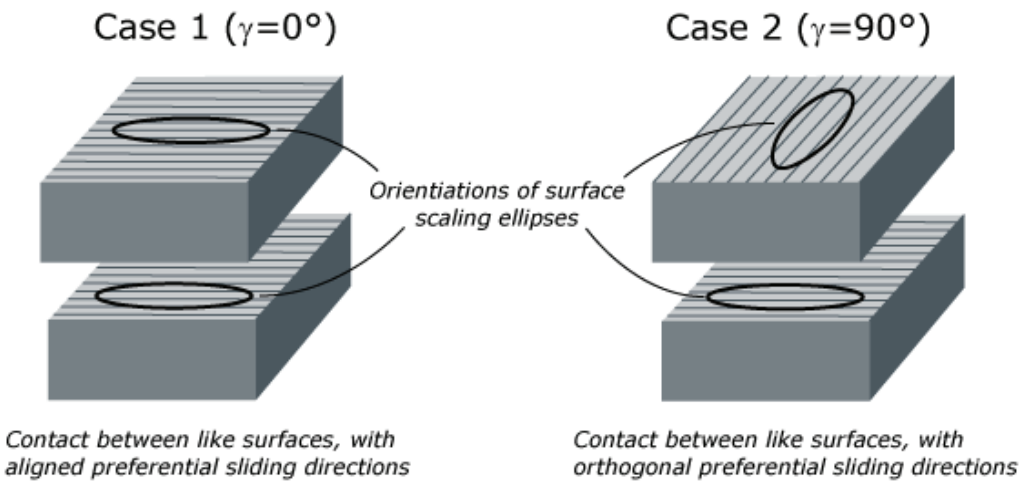


Figure 10: Examples of contact between like surfaces with directional preferences aligned (Case 1) and orthogonal (Case 2).

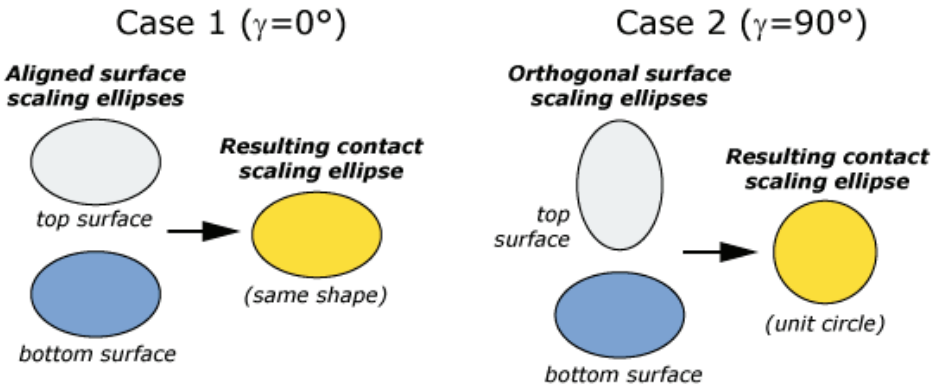


Figure 11: Scaling ellipse combinations when directional preferences of like surfaces are aligned (Case 1) and orthogonal (Case 2).

Input File Usage: Use the following option to specify the nominal friction coefficient for an anisotropic friction model with surface-based directional preferences:

**FRICTION*, ANISOTROPIC BEHAVIOR=SURFACE PROPERTY

Use the following option to specify the surface-based directional preferences and the frictional directional preference factor:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=ORIENTATION

Abaqus/CAE Usage: Defining an anisotropic friction model in Abaqus/Explicit is not supported in Abaqus/CAE.

Weighting Methods for Combining Preferential Direction Effects of Surfaces for Anisotropic Friction

Abaqus/Explicit provides three options for computing weight factors for determination of $s_{combined}(\theta) = w_1 s_{surf1}(\theta) + w_2 s_{surf2}(\theta)$. $w_1 = w_2 = \frac{1}{2}$ for all cases with $\epsilon_{surf1} = \epsilon_{surf2}$. For cases with $\epsilon_{surf1} \neq \epsilon_{surf2}$, w_1 is established according to one of the following three weighting methods (and $w_2 = 1 - w_1$):

1. Balanced weighting, in which $w_1 = \frac{1}{2}$.
2. ϵ_s-proportional weighting, in which $w_1 = \frac{\epsilon_{surf1}}{\epsilon_{surf1}+\epsilon_{surf2}}$.
3. Maximum-ϵ_s-dominant weighting, in which $w_1 = 1$ if $\epsilon_{surf1} > \epsilon_{surf2}$ and $w_1 = 0$ if $\epsilon_{surf1} < \epsilon_{surf2}$.

Consider a contact interaction in which only one surface introduces directional preference, with $\epsilon_{surf1} = \frac{1}{3}$ and $\epsilon_{surf2} = 0$. In this case balanced weighting leads to $\epsilon_{combined} = \frac{1}{6}$, as shown in *Figure 12*. The other two weighing methods lead to $w_1 = 1$ and $w_2 = 0$ for this example, in which case the combined scaling ellipse is identical to the scaling ellipse for the surface with directional preference and $\epsilon_{combined} = \frac{1}{3}$.

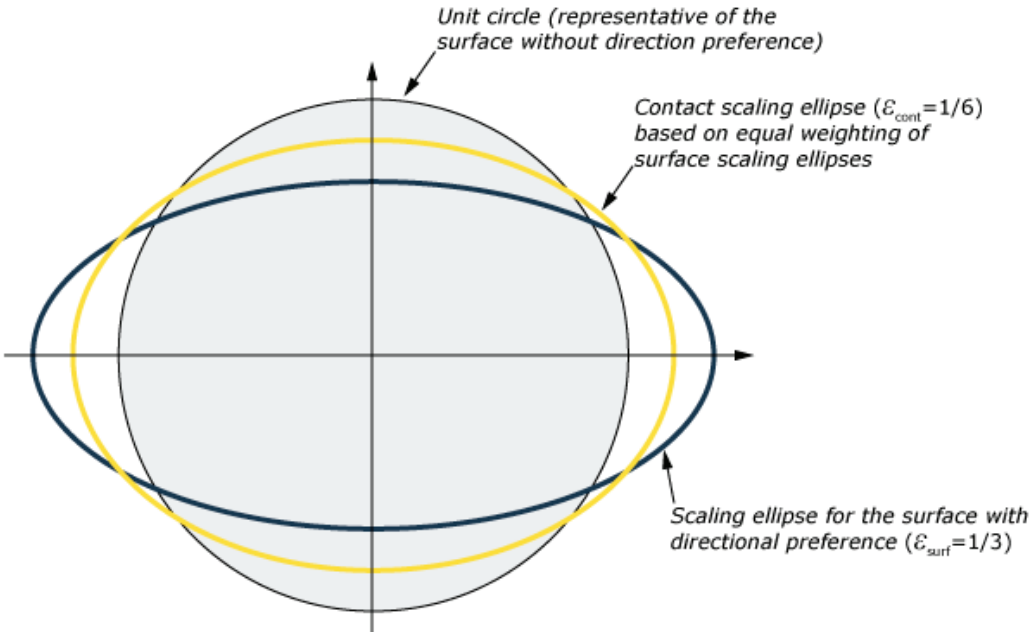


Figure 12: Balanced weighting for anisotropic friction with one surface contributing directional preference.

The effect of balanced weighting for a situation with $\epsilon_{surf1} = \frac{1}{3}$ and $\epsilon_{surf2} = \frac{1}{5}$ with a 45° angle between major axes of the surface scaling ellipses is represented in *Figure 13*. Weight factors for this combination of surface scaling ellipses according to the different weighting methods are:

- With balanced weighting: $w_1 = \frac{1}{2}$ and $w_2 = \frac{1}{2}$.
- With ϵ_s-proportional weighting: $w_1 = \frac{5}{8}$ and $w_2 = \frac{3}{8}$. With this weighting, the contact scaling ellipse would be closer to the scaling ellipse for surface 1 than the contact scaling ellipse for balanced weighting.

- With maximum-ϵ_s-dominant weighting: $w_1 = 1$ and $w_2 = 0$. With this weighting, the contact scaling ellipse would be identical to the scaling ellipse for surface 1.

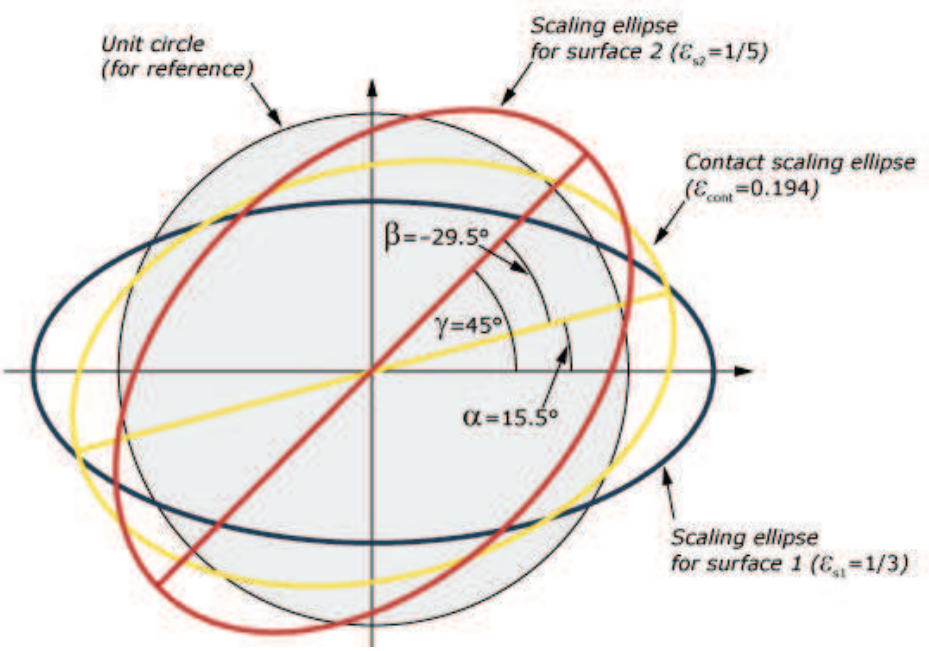


Figure 13: Balanced weighting for anisotropic friction with both surfaces contributing directional preference.

Input File Usage: Use the following option to specify the balanced weighting method for combining surface-based directional preferences for the anisotropic friction model (default):

**COMBINATORIAL RULE*, ANISOTROPY WEIGHTING=BALANCED

Use the following option to specify the epsilon-proportional weighting method for combining surface-based directional preferences for the anisotropic friction model:

**COMBINATORIAL RULE*, ANISOTROPY WEIGHTING=PROPORTIONAL

Use the following option to specify the maximum-epsilon-dominant weighting method for combining surface-based directional preferences for the anisotropic friction model:

**COMBINATORIAL RULE*, ANISOTROPY WEIGHTING=DOMINANT

Abaqus/CAE Usage: Defining an anisotropic friction model in Abaqus/Explicit is not supported in Abaqus/CAE.

Defining the Friction Coefficient

As explained, the critical shear stress surface is proportional to the average, or nominal, friction coefficient. For convenience, you can also specify the minimum or maximum friction coefficient rather than the average one. Internally the value will be converted to the average friction coefficient using the following formulas:

$$\mu_{\min} = [1 - (\omega_1 \epsilon_1 + \omega_2 \epsilon_2)]\, \mu_{nom},$$
$$\mu_{\max} = [1 + (\omega_1 \epsilon_1 + \omega_2 \epsilon_2)]\, \mu_{nom}.$$

Input File Usage: Use the following option to specify the average friction coefficient (default):

**FRICTION*, ANISOTROPIC BEHAVIOR=SURFACE PROPERTY, NOMINAL=AVERAGE

Use the following option to specify the minimum friction coefficient:

**FRICTION*, ANISOTROPIC BEHAVIOR=SURFACE PROPERTY, NOMINAL=MINIMUM

Use the following option to specify the maximum friction coefficient:

**FRICTION*, ANISOTROPIC BEHAVIOR=SURFACE PROPERTY, NOMINAL=MAXIMUM

Abaqus/CAE Usage: Defining an anisotropic friction model in Abaqus/Explicit is not supported in Abaqus/CAE.

Measure of the Eccentricity of the Scaling Ellipse

The measure of eccentricity of the scaling ellipse for each surface is defined using the frictional directional preference factor ϵ. Sometimes it is useful to specify the scaling ellipse using the ratio of the friction coefficients:

$$r = \frac{\mu_{\max}}{\mu_{\min}}.$$

The relationship between ϵ and r is given by the following formula:

$$\epsilon = \frac{(r-1)}{(r+1)}.$$

Input File Usage: To specify the frictional directional preference factor ϵ (default):

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=ORIENTATION, FRICTION ANISOTROPY=EPSILON

To specify the ratio of the friction coefficients r:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=ORIENTATION, FRICTION ANISOTROPY=RATIO

Abaqus/CAE Usage: Defining an anisotropic friction model in Abaqus/Explicit is not supported in Abaqus/CAE.

Output of the Preferred Directions

The preferred local directions for each surface in contact can be output by requesting the generic output variable CORIENT (with the respective vector components CORIENT1 and CORIENT2).

Anisotropic Friction with Directional Preference Associated with Contact Orientation

Directional preference for this anisotropic friction model is specified as a contact property and is implicitly associated with one surface of a contact interaction. This friction model is available in Abaqus/Standard and is less general than the anisotropic friction model discussed in *Anisotropic Friction with Directional Preference*

as a Surface Property, which allows both surfaces of a contact interaction to contribute directional preference characteristics.

If you indicate that this anisotropic friction model should be used, you must specify two friction coefficients, where μ_1 is the coefficient of friction in the first local tangent direction and μ_2 is the coefficient of friction in the second local tangent direction. The critical contact shear stress surface for this friction model (see *Figure 14*) is elliptical in the τ_1–τ_2 plane, like the anisotropic friction model discussed in *Anisotropic Friction with Directional Preference as a Surface Property*. The shape of the critical contact shear stress surface remains constant in this case, with the extreme points being $\tau_1^{crit} = \mu_1 p$ and $\tau_2^{crit} = \mu_2 p$.

The orientation of the critical contact shear stress for this friction model evolves with the local tangent directions of the contact interaction, which are discussed in *Local Tangent Directions on a Surface*. For example, local tangent directions for the finite-sliding, surface-to-surface contact formulation are established from, and evolve with, the secondary surface of the contact interaction, so, in this example, directional preference for this anisotropic friction model would evolve with the orientation of the secondary surface (and would be independent of the main surface).

The size of the critical-shear-stress ellipse will change with the change in contact pressure between the surfaces.

The direction of slip, $d\gamma_\alpha$, is orthogonal to the critical shear stress surface.

The optional equivalent shear stress limit, τ_{max}, is applied to the *scaled* equivalent shear stress, $\bar{\tau}_{eq}$, for anisotropic friction. See *Anisotropic friction* for the definition and discussion of $\bar{\tau}_{eq}$.

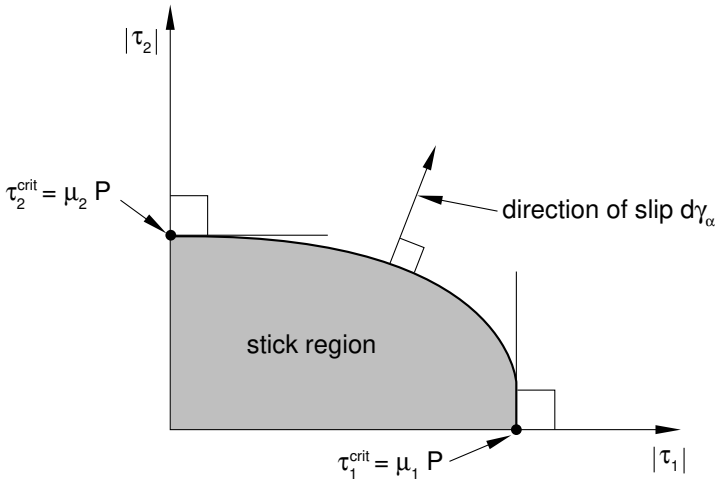


Figure 14: Critical shear stress surface for the anisotropic friction model.

The friction coefficients can depend on slip rate, contact pressure, temperature, and field variables. By default, it is assumed that the friction coefficients do not depend on field variables.

Input File Usage: **FRICTION*, ANISOTROPIC BEHAVIOR=LEGACY, DEPENDENCIES=*n*

μ_1, μ_2, $\dot{\gamma}_{eq}$, p, $\bar{\theta}$, $\bar{f}^{\alpha}$

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Tangential Behavior**: **Friction formulation: Penalty**: **Friction**, **Directionality: Anisotropic**

If necessary, toggle on **Use slip-rate-dependent data**, **Use contact-pressure-dependent data**, and/or **Use temperature-dependent data**; and/or specify the **Number of field variable** dependencies in addition to slip rate, contact pressure, and temperature.

Preventing Slipping regardless of Contact Pressure

Abaqus offers the option of specifying an infinite coefficient of friction ($\mu = \infty$). This type of surface interaction is called "rough" friction, and with it all relative sliding motion between two contacting surfaces is prevented (except for the possibility of "elastic slip" associated with penalty enforcement) as long as the corresponding normal-direction contact constraints are active. In most cases Abaqus/Standard uses a penalty method to enforce these tangential constraints; however, a Lagrange multiplier method is used during general (non-perturbation) analysis steps if the corresponding normal-direction constraints have directly enforced "hard contact" or exponential pressure-overclosure behavior. Abaqus/Explicit uses either a kinematic or penalty method, depending on the contact formulation chosen.

Rough friction is intended for nonintermittent contact; once surfaces close and undergo rough friction, they should remain closed. Convergence difficulties might arise in Abaqus/Standard if a closed contact interface with rough friction opens, especially if large shear stresses have developed. The rough friction model is typically used in conjunction with the no separation contact pressure-overclosure relationship for motions normal to the surfaces (see *Using the No Separation Relationship*), which prohibits separation of the surfaces once they are closed.

When rough friction is used with the no separation relationship for hard contact in Abaqus/Explicit specified with the kinematic contact method, no relative motions of the surfaces will occur. For hard contact in Abaqus/Explicit specified with the penalty contact method, relative motions will be limited to the elastic slip and penetration corresponding to the inexact satisfaction of the contact constraints by the applied penalty forces. When softened tangential behavior is specified in Abaqus/Explicit (see *Defining Tangential Softening in Abaqus/Explicit* below), the relative surface motions will be governed by the specified softening behavior.

Input File Usage: **FRICTION*, ROUGH

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical**->**Tangential Behavior**: **Friction formulation: Rough**

Shear Stress Versus Elastic Slip While Sticking

In some cases some incremental slip might occur even though the friction model determines that the current frictional state is "sticking." In other words, the slope of the shear (frictional) stress versus total slip relationship might be finite while in the "sticking" state, as shown in *Figure 15*.

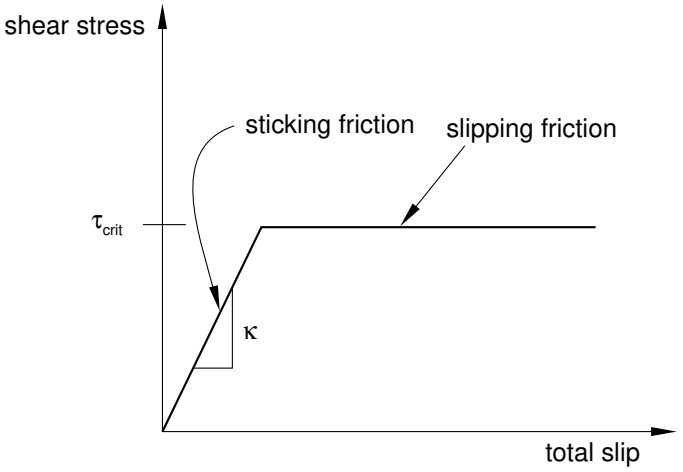


Figure 15: Elastic slip versus shear traction relationship for sticking and slipping friction.

The relationship shown in this figure is analogous to elastic-plastic material behavior without hardening: κ corresponds to Young's modulus, and τ_{crit} corresponds to yield stress; sticking friction corresponds to the elastic regime, and slipping friction corresponds to the plastic regime. A finite value of the sticking stiffness might reflect a user-specified physical behavior or might be characteristic of the constraint enforcement method.

Frictional constraints are enforced with a stiffness (penalty method) by default in Abaqus/Standard and for the general contact algorithm in Abaqus/Explicit; in this case the sticking stiffness will have a finite value. An infinite sticking stiffness, in which case the elastic slip is always zero, can be achieved with the optional Lagrange multiplier method for imposing frictional constraints in Abaqus/Standard or with the kinematic constraint method (available only for contact pairs) in Abaqus/Explicit. In Abaqus/Explicit some tangential contact damping acts on the elastic slip rate by default, as discussed in *Contact Damping*. Tangential softening to reflect a physical behavior is available only in Abaqus/Explicit.

Defining Tangential Softening in Abaqus/Explicit

To activate softened tangential behavior in Abaqus/Explicit, specify the slope of the shear stress versus elastic slip relationship (κ in *Figure 15*). User subroutine *VFRIC* cannot be used in conjunction with softened tangential behavior.

Input File Usage: **FRICTION*, SHEAR TRACTION SLOPE=κ

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical**->**Tangential Behavior**: **Friction formulation: Penalty** or **Static-Kinetic**
Exponential Decay: **Elastic Slip**, **Specify:** κ

Stiffness Method for Imposing Frictional Constraints in Abaqus/Standard

The stiffness method used for friction in Abaqus/Standard is a penalty method that permits some relative motion of the surfaces (an “elastic slip”) when they should be sticking (similar to the allowable elastic slip defined with softened tangential behavior in Abaqus/Explicit). While the surfaces are sticking (that is, $\tau_{eq} < \tau_{crit}$ for the basic isotropic Coulomb friction), the magnitude of sliding is limited to this elastic slip. Abaqus continually adjusts the magnitude of the penalty constraint to enforce this condition.

The stiffness method in Abaqus/Standard requires the selection of an allowable elastic slip, γ_{crit}. Using a large γ_{crit} in the simulation makes convergence of the solution more rapid at the expense of solution accuracy (there is greater relative motion of the surfaces when they should be sticking). Behavior in which no slip is permitted in the sticking state is approximated more accurately by allowing only a small γ_{crit}. If γ_{crit} is chosen very small, convergence problems may occur; in that case, it might be better to use the Lagrange multiplier method to apply the sticking constraint (see *Lagrange Multiplier Method for Imposing Frictional Constraints in Abaqus/Standard* later in this section).

The default value of allowable elastic slip used by Abaqus/Standard generally works very well, providing a conservative balance between efficiency and accuracy. Abaqus/Standard calculates γ_{crit} as a small fraction of the “characteristic contact surface length,” $\overline{l}_{char}$, and scans all the facets of all the secondary surfaces when calculating $\overline{l}_{char}$. Abaqus/Standard reports the value of $\overline{l}_{char}$ used for each contact pair in the data (`.dat`) file if you request detailed printout of contact constraint information (see *Controlling the Amount of analysis input file processor Information Written to the Data File*). The allowable elastic slip is given as $\gamma_{crit} = F_f \overline{l}_{char}$, where F_f is the slip tolerance; the default value of F_f is 0.005.

This method of calculating the default allowable elastic slip is used for all analysis procedures in Abaqus/Standard except steady-state transport analysis (*Steady-State Transport Analysis*), in which the penalty constraint is based on a maximum allowable slip rate, $\dot{\gamma}_{crit}$. The maximum slip rate is calculated as

$$\dot{\gamma}_{crit} = F_f 2\omega R,$$

where ω is the angular spinning rate and R is the radius of the rolling structure.

If the stiffness method is used for an anisotropic friction model, γ_{crit} is a nominal allowable elastic slip (or slip rate). If γ_1^e and γ_2^e represent components of elastic slip in the t_1 and t_2 tangent directions, respectively, the transition from stick to slip will occur when $\gamma_{eff} = \gamma_{crit}$, where γ_{eff} is computed as

$$\gamma_{eff} = \sqrt{\frac{2\left[\left(\mu_1\gamma_1^e\right)^2+\left(\mu_2\gamma_2^e\right)^2\right]}{\mu_1{}^2+\mu_2{}^2}}.$$

For example, if $\gamma_2^e = 0$, the stick/slip transition will occur at $\gamma_1^e = \sqrt{\frac{1}{2}\left(1+\frac{\mu_2^2}{\mu_1^2}\right)}\gamma_{crit}$. Depending on values of μ_1 and μ_2, this γ_1^e can be greater or smaller than γ_{crit}. As another example, if the "1" and "2" components of elastic slip are equal, the stick/slip transition will occur at $\gamma_1 = \gamma_2 = \frac{\gamma_{crit}}{\sqrt{2}}$, such that the magnitude of the elastic slip, $\gamma_{mag} = \sqrt{\gamma_1^2+\gamma_2^2} = \gamma_{crit}$.

Cases in Which the Default Elastic Slip Value May Not Be Suitable

In certain situations the default value for the allowable elastic slip might not be suitable. For example, secondary surfaces defined by node-based surfaces or some contact element types, such as GAPUNI elements, have no physical dimensions and Abaqus/Standard cannot estimate a value of $\overline{l}_{char}$. For models containing only node-based surfaces or these types of contact elements, Abaqus/Standard first tries to use the "characteristic contact surface length" of the other contact pairs in the model. If there are none, it calculates $\overline{l}_{char}$ using all the elements in the model and issues a warning message. If a model contains no elements for which a characteristic length can be determined (for example, if it contains only substructures), Abaqus/Standard has no information with which to calculate $\overline{l}_{char}$. As a result, it uses a value of 1.0 and issues a warning message. If the contact surface face dimensions vary greatly, the average value of $\overline{l}_{char}$ might be unreasonable for some contact surfaces. The elastic slip should then be specified directly for the surfaces with a much smaller "characteristic face dimension."

There are two approaches to modify the allowable elastic slip. One approach is to specify γ_{crit} directly; the other is to specify the slip tolerance, F_f. Some analyses call for nondefault γ_{crit} or F_f only in specific steps (see *Changing Friction Properties during an Abaqus/Standard Analysis* above).

Specifying the Allowable Elastic Slip Directly

You can modify the default allowing elastic slip distance by specifying the absolute magnitude of γ_{crit} in units of length directly or by specifying an unitless value. Specify a reasonable value for the relative displacement that might occur before surfaces actually begin to slip. Typically, the allowable elastic slip is set to a small fraction (10^{-2}–10^{-4}) of a "characteristic contact surface face dimension." In a steady-state transport analysis you can define the maximum allowable viscous slip rate, $\dot{\gamma}_{crit}$.

The specified allowable elastic slip is used only for surface pairings that reference a contact property definition that contains a friction definition. Otherwise, the allowable elastic slip is ignored.

Input File Usage: Use both of the following options to change the default slip tolerance:

**CONTACT*
**CONTACT CONTROLS ASSIGNMENT*, TYPE=FRICTION ENFORCEMENT
surface_1, surface_2, PENALTY, SLIP TOLERANCE DISTANCE, γ_{crit}

Abaqus/CAE Usage: Defining the slip tolerance using contact controls is not supported in Abaqus/CAE.

Controlling the Allowable Elastic Slip as a Fraction of the Characteristic Facet Dimension

You can alter the default value of the slip tolerance, F_f. This method of altering the default elastic slip is convenient if the goal is to increase computational efficiency (in which case a value larger than the default of 0.005 would be given) or if the goal is to increase accuracy (in which case a value smaller than the default would be given).

Input File Usage: Use both of the following options to change the default slip tolerance:

**CONTACT*
**CONTACT CONTROLS ASSIGNMENT*, TYPE=FRICTION ENFORCEMENT
surface_1, surface_2, PENALTY, UNITLESS SLIP TOLERANCE, F_f

Abaqus/CAE Usage: Defining the slip tolerance through contact controls is not supported in Abaqus/CAE.

Alternative Method for Modifying Elastic Slip

The alternative method to modify the elastic slip tolerance is to use the friction property directly. In this case any contact in the model associated with a particular friction property will have the specified elastic slip. This method can be useful for cases where the physical contact properties and the best choice of elastic slip closely align for the same regions of contact.

You can provide the absolute magnitude of γ_{crit} directly. For more information, see *Specifying the Allowable Elastic Slip Directly*.

The specified allowable elastic slip is used only for the contact pairs referencing the contact property definition that contains the friction definition. For example, three surfaces `ASURF`, `BSURF`, and `CSURF` form two contact pairs that each refer to their own contact property definition, as shown below.

Contact Pair	Contact Property	γ_{crit}
`ASURF, BSURF`	`DEFAULT`	$F_f \bar{l}_{char}$
`CSURF, BSURF`	`NONDEF`	0.1

In the `DEFAULT` contact property definition no value for γ_{crit} is specified; therefore, the allowable elastic slip used for the friction interaction between `ASURF` and `BSURF` would be the default value $F_f \bar{l}_{char}$. In the `NONDEF` contact property definition a value of 0.1 is specified for γ_{crit}, which will be the allowable elastic slip used for the friction interaction between `CSURF` and `BSURF`.

You can alter the default value of the slip tolerance, F_f. For more information, see *Controlling the Allowable Elastic Slip as a Fraction of the Characteristic Facet Dimension*.

Input File Usage: **FRICTION*, ELASTIC SLIP=γ_{crit}

To specify the slip tolerance, use the following option:

**FRICTION*, SLIP TOLERANCE=F_f

Abaqus/CAE Usage: Interaction module: Contact property editor: **Mechanical->Tangential Behavior**: **Friction formulation: Penalty** or **Static-Kinetic**
Exponential Decay: **Elastic Slip**, **Absolute distance:** γ_{crit}

To specify the slip tolerance, use the following option:

Interaction module: contact property editor: **Mechanical->Tangential Behavior**: **Friction formulation: Penalty** or **Static-Kinetic**
Exponential Decay: **Elastic Slip**, **Fraction of characteristic surface dimension:** F_f

Stiffness Method for Imposing Frictional Constraints in Abaqus/Explicit

The stiffness method used for friction with the general contact algorithm in Abaqus/Explicit and, optionally, with the contact pair method in Abaqus/Explicit is a penalty method that permits some relative motion of the surfaces (an "elastic slip") when they should be sticking (similar to the allowable elastic slip defined with softened tangential behavior in Abaqus/Explicit). While the surfaces are sticking (that is, $\tau_{eq} < \tau_{crit}$), the magnitude of sliding is limited to this elastic slip. Abaqus continually adjusts the magnitude of the penalty constraint to enforce this condition.

In Abaqus/Explicit you can choose to have contact constraints for the contact pair algorithm enforced with the penalty method; the general contact algorithm always uses a penalty method (see *Contact Constraint Enforcement Methods in Abaqus/Explicit*).

The default penalty stiffness for frictional constraints is chosen automatically by Abaqus/Explicit and is the same as would be used for normal hard contact constraints. Softening in the normal direction does not affect the penalty stiffness used to enforce stick conditions. If tangential softening is specified (see *Defining Tangential Softening in Abaqus/Explicit* above), the penalty stiffness will be equal to the value specified for the slope of the shear stress versus elastic slip relationship. You can specify a scale factor to adjust the penalty stiffness, as discussed in *Contact Controls for General Contact in Abaqus/Explicit* and *Contact Controls for Contact Pairs in Abaqus/Explicit*.

Lagrange Multiplier Method for Imposing Frictional Constraints in Abaqus/Standard

In Abaqus/Standard the sticking constraints at an interface between two surfaces can be enforced exactly by using the Lagrange multiplier implementation. With this method there is no relative motion between two closed surfaces until $\tau_{eq} = \tau_{crit}$. However, the Lagrange multipliers increase the computational cost of the analysis by adding more degrees of freedom to the model and often by increasing the number of iterations required to obtain a converged solution. The Lagrange multiplier formulation might even prevent convergence of the solution, especially if many points are iterating between sticking and slipping conditions. This effect can occur particularly if locally there is a strong interaction between slipping/sticking conditions and contact stresses.

Because of the added cost of using the Lagrange friction formulation, it should be used only in problems where the resolution of the stick/slip behavior is of utmost importance, such as modeling fretting between two bodies. In typical metal forming applications or for contact of rubber components, accurate resolution of the stick/slip behavior is not important enough to justify the added costs of the Lagrange multiplier formulation.

Alternatively, you can assign the Lagrange multipler method for imposing friction constraints by using the friction property directly. In this method any contact surfaces in the model associated with a particular friction property have the Lagrange multiplier method assigned. This method is not preferred; however, it can be useful for cases where the physical contact properties and the best choice of imposing friction constraints closely align for the same regions of contact.

Input File Usage: Use both of the following options to change the default slip tolerance:

**CONTACT*
**CONTACT CONTROLS ASSIGNMENT*, TYPE=FRICTION ENFORCEMENT
surface_1, surface_2, LAGRANGE

Use the following option to specify the friction property directly:

**FRICTION*, LAGRANGE

Abaqus/CAE Usage: Defining the slip tolerance through contact controls is not supported in Abaqus/CAE.

Use the following option to specify the friction property directly:

Interaction module: contact property editor: **Mechanical->Tangential Behavior**: **Friction formulation: Lagrange Multiplier**

Kinematic Method for Imposing Frictional Constraints in Abaqus/Explicit

By default, the contact pair algorithm in Abaqus/Explicit uses a kinematic method for imposing frictional constraints (see *Contact Constraint Enforcement Methods in Abaqus/Explicit*). The kinematic method applies sticking constraints in a way similar to the optional Lagrange multiplier method in Abaqus/Standard; however, the algorithm is quite different. The value of the force required to enforce sticking at a node is first calculated using the mass associated with the node; the distance the node has slipped; the time increment; and additionally for softened contact, the current value of the elastic slip and the elastic slip versus shear stress slope. For hard contact this sticking force is that which is required to maintain the node's position on the opposite surface in the predicted configuration. For softened contact this force is consistent with the user-specified value for the slope of the shear stress versus elastic slip relationship. The sticking force for each node is calculated using the mass associated with the node, the distance the node has slipped, the shear traction-elastic slip slope (if softened contact is specified in the tangential direction), and the time increment. If the shear stress at the node calculated using this force is less than τ_{crit}, the node is considered to be sticking and this force is applied to each surface in opposing directions. If the shear stress exceeds τ_{crit}, the surfaces are slipping and the force corresponding to τ_{crit} is applied. In either case the forces result in acceleration corrections tangential to the surface at the secondary node and either the nodes of the main surface facet or the points on the analytical rigid surface that it contacts.

User-Defined Friction Model

You can define the shear stress between contacting surfaces through a user subroutine when the friction behavior provided by Abaqus is not sufficient. The shear stress can be defined as a function of a number of variables such as slip, slip rate, temperature, and field variables. You can also introduce a number of solution-dependent state variables that you can update and use within the friction user subroutines. You can declare a number of properties or constants associated with your friction model and use these values in the user subroutine.

In Abaqus/Standard, when user-defined friction is specified in procedures with temperature unknowns, the temperatures passed into the user subroutines correspond to values at the end of an increment at the secondary node and the corresponding point on the main. However, there are situations where not all nodes in the contact constraint connectivity have temperature degrees of freedom, such as in the case where only one of the secondary

or main surfaces is meshed with elements with temperature degrees of freedom while the other surface is not. In such situations, contact only enforces mechanical constraints, and the temperature values passed into the user subroutines are not temperature-solution variables at the end of an increment but rather based on boundary conditions and initial conditions (if present) in that order of precedence.

In addition to the friction user subroutines, subroutines are available for defining the complete mechanical interaction between surfaces, including the interaction in the normal direction as well as the frictional behavior in the tangential direction; see *User-Defined Interfacial Constitutive Behavior* for more information.

Defining Generic Frictional Behavior

You can define a generic frictional behavior between contacting surfaces using user subroutine *FRIC* in Abaqus/Standard. In Abaqus/Explicit the generic frictional behavior for contact pairs is defined in user subroutine *VFRIC*, while the generic frictional behavior for general contact is defined in user subroutine *VFRICTION*.

Input File Usage: Use the following option to define a frictional behavior with user subroutine *FRIC* or *VFRIC*:

**FRICTION*, USER, DEPVAR=*n*, PROPERTIES=*p*

Use the following option to define a frictional behavior with user subroutine *VFRICTION*:

**FRICTION*, USER=FRICTION, DEPVAR=*n*, PROPERTIES=*p*

Abaqus/CAE Usage: Use the following options to define a frictional behavior with user subroutine *FRIC* or *VFRIC*:

Interaction module: contact property editor: **Mechanical->Tangential Behavior**: **Friction formulation: User-defined**, **Number of state-dependent variables:** *n*, **Friction Properties**

User subroutine *VFRICTION* is not supported in Abaqus/CAE.

Defining Complex Friction Coefficient Dependencies

Abaqus provides a simple way to specify complex dependence of friction coefficients with user subroutines *FRIC_COEF* (Abaqus/Standard) and *VFRIC_COEF* (Abaqus/Explicit). *VFRIC_COEF* can be used only with general contact. These user subroutines have a much narrower scope and are much simpler to create than user subroutines that control all aspects of a friction model (*FRIC* and *VFRIC*, discussed in *Defining Generic Frictional Behavior*). In addition, user subroutine *FRIC_COEF* preserves heuristics built into Abaqus/Standard friction algorithms to assist convergence behavior; many of these built-in heuristics are bypassed with user subroutine *FRIC*.

FRIC_COEF can be used for isotropic or anisotropic friction behavior. Friction coefficients can depend on contact pressure, temperature, and a number of contact slip–related variables at the current time. *VFRIC_COEF* is limited to controlling a single friction coefficient per contact constraint for isotropic friction or for the anisotropic friction model discussed in *Anisotropic Friction with Directional Preference as a Surface Property*. In addition to temperature and contact pressure dependence, the friction coefficient can also depend on equivalent contact slip and contact slip rates.

Input File Usage: **FRICTION*, USER=COEFFICIENT, PROPERTIES=*p*

Abaqus/CAE Usage: User subroutines *FRIC_COEF* and *VFRIC_COEF* are not supported in Abaqus/CAE.

Consideration of Incremental Rotation of Shell and Beam Thickness Offsets in Abaqus/Explicit

By default, in Abaqus/Explicit slip increment calculations for friction do not account for the incremental rotation of shell and beam thickness offsets, and frictional constraints do not apply a moment to nodes offset from the contact interface due to shell or beam thicknesses. This behavior can be modified for general contact; for details see *Consideration of Shell and Beam Thickness Offsets*.

Improving Abaqus/Standard Simulations That Include Friction in the Surface Interactions

Several features of the frictional interaction of surfaces can have a strong influence on the rate of convergence in an Abaqus/Standard simulation.

Unsymmetric Terms in the System of Equations

Friction constraints produce unsymmetric terms when the surfaces are sliding relative to each other. These terms have a strong effect on the convergence rate if frictional stresses have a substantial influence on the overall displacement field and the magnitude of the frictional stresses is highly solution dependent. Abaqus/Standard will automatically use the unsymmetric solution scheme if $\mu > 0.2$ or if μ is pressure-dependent. If desired, you can turn off the unsymmetric solution scheme; see *Matrix Storage and Solution Scheme in Abaqus/Standard*.

No slip occurs with rough friction; the contribution to the stiffness will be fully symmetric, and Abaqus/Standard will use the symmetric solution scheme by default.

Heat Generated by Frictional Interaction of Surfaces

In fully coupled temperature-displacement analysis and fully coupled thermal-electrical-structural analysis, all dissipated mechanical (frictional) energy is converted to heat and distributed equally between the two surfaces by default. This behavior can be modified; for details about this and other thermal surface interactions, see *Thermal Contact Properties*.

Temperature and Field-Variable Dependence of Friction Properties for Structural Elements

Temperature and field-variable distributions in beam and shell elements can generally include gradients through the cross-section of the element. Contact between these elements occurs at the reference surface; therefore, temperature and field-variable gradients in the element are not considered when determining friction properties that depend on these variables.

Surface Interaction Variables Related to Friction

Abaqus provides output of the shear stresses at points on the secondary surface that use a surface interaction model containing frictional properties. The shear stresses, CSHEAR1 and CSHEAR2, are given in the two orthogonal local tangent directions, which are constructed on the main surface (see *Contact Formulations in Abaqus/Standard*). There is only one local tangent direction in two-dimensional problems. Details about how to request contact surface variable output are given in *About Contact Pairs in Abaqus/Standard* and *About Contact Pairs in Abaqus/Explicit*.

Contour plots of these variables can also be plotted in Abaqus/CAE.

References

- Oden, J. T., and J. A. C. Martins, "Models and Computational Methods for Dynamic Friction Phenomena," *Computer Methods in Applied Mechanics and Engineering*, vol. 52, pp. 527–634, 1985.

User-Defined Interfacial Constitutive Behavior

Products: Abaqus/Standard Abaqus/Explicit

References:

- *UINTER*
- *VUINTER*
- *VUINTERACTION*
- **SURFACE INTERACTION*

Overview

User-defined interfacial constitutive behavior:

- is provided so that any constitutive behavior across an interface can be added to the library of existing models such as softened contact and Coulomb friction;
- requires that a constitutive model (or a library of models) for the interface be programmed in user subroutine *UINTER* in Abaqus/Standard;
- requires that a constitutive model (or a library of models) for the interface be programmed in user subroutine *VUINTER* in Abaqus/Explicit when using the contact pair algorithm;
- requires that a constitutive model (or a library of models) for the interface be programmed in user subroutine *VUINTERACTION* in Abaqus/Explicit when using the general contact algorithm;
- is available only for surface-based contact definition involved in stress/displacement, coupled temperature-displacement, coupled thermal-electrical-structural, or heat transfer analysis; and
- requires considerable effort and expertise: the feature is very general and powerful, but it is intended for advanced users.

Purpose of User Subroutines `UINTER`, `VUINTER`, and `VUINTERACTION`

User subroutines *UINTER*, *VUINTER*, and *VUINTERACTION* provide a very general interface for you to define the constitutive behavior across the interface between two surfaces. These subroutines replace all built-in interfacial constitutive behavior models; hence, no other contact property definitions (e.g., friction, thermal conductance, etc.) can be specified in conjunction with them.

In a stress/displacement analysis you must define the stresses, both normal and tangential, at the secondary node (or points on the secondary surface) at the current point in time. In a coupled temperature-displacement analysis and a coupled thermal-electrical-structural analysis you must also define the heat flux across the interface. The constitutive calculation thus involves computing the stresses and heat fluxes based on the increments in relative position of the secondary node with respect to the main surface (which act as strains in this context), temperature at the surface, and predefined field variables. The calculations would typically involve solution-dependent state variables, which can be updated inside these routines. If contact damping is to be included in the interfacial constitutive model, you must include the damping contribution in the stress definition.

When a user subroutine is used to define the interfacial constitutive behavior, all decisions regarding the contact status of a secondary node must be made inside the subroutine based on the information provided. You can make such decisions based on the values of the relative position of the point on the secondary surface with respect to the main surface and appropriately defined solution-dependent state variables. Thus, usage of this feature involves

not only developing a constitutive behavior of the interface but also developing conditions under which contact is active at a given point on the secondary surface. The interface is always assumed to be massless.

User subroutine *UINTER* will be called for each contact constraint location of affected contact pairs in each iteration of an Abaqus/Standard analysis. The input to this user subroutine includes the current relative position of a particular constraint point on the secondary surface with respect to the corresponding closest point on the main surface, as well as the incremental relative motion between these two points. Values of temperature and field variables at the constraint point on the secondary surface and the corresponding closest point on the main surface and several other variables are also provided as input. In addition to defining the contact stress or heat flux, appropriate Jacobian terms must also be defined to ensure proper convergence characteristics in Abaqus/Standard.

User subroutine *VUINTER* will be called multiple times for the affected contact pairs in each time increment of an Abaqus/Explicit analysis. All secondary nodes are processed in each call to *VUINTER*, whereas only a single constraint is processed in each call to *UINTER*. Similar input is provided to *VUINTER* as *UINTER*.

User subroutine *VUINTERACTION* will be called multiple times for each interacting surface in each time increment of an Abaqus/Explicit analysis. Points of potential contact for a given interaction are processed in blocks in calls to *VUINTERACTION*. Similar input is provided to *VUINTERACTION* as *VUINTER*.

Interfacial Constants

You must specify the number of interfacial constants that are needed in user subroutine *UINTER*, *VUINTER*, or *VUINTERACTION*; and you must provide values for all these constants. All surface constitutive behavior calculations and all decisions regarding the contact status at a secondary node (or a point on the secondary surface in question) must be programmed in the user subroutine. Any other contact property definitions included in the analysis are reported as an error.

Input File Usage: For contact interactions defined through user subroutine *UINTER* or *VUINTER*:

**SURFACE INTERACTION*, USER, PROPERTIES=*number_of_material_constants*

For contact interactions defined through user subroutine *VUINTERACTION*:

**SURFACE INTERACTION*, USER=INTERACTION,
PROPERTIES=*number_of_material_constants*

Tracking Thickness When `VUINTER` or `VUINTERACTION` Is Used

Computations to determine the precise distance to a main surface are avoided if a secondary node can be easily determined to be separated from the main surface, accounting for all thickness effects and built-in contact property models, by more than a threshold distance called the tracking thickness. Abaqus/Explicit uses an internal default value for the tracking thickness. If a built-in contact property model is in effect, the tracking thickness is quite small to help reduce computation time. However, if user subroutine *VUINTER* or user subroutine *VUINTERACTION* is in effect, the default tracking thickness is infinite so that all secondary nodes are supplied to the user subroutines as having potential interactions. Alternatively, you can specify the tracking thickness in conjunction with a user-defined surface interaction model. In this case secondary nodes whose proximity to their main surfaces are within this thickness are available for user-defined interactions. Use of a user-specified tracking thickness is supported only with node-to-surface contact and not with edge-to-edge contact.

Input File Usage: **SURFACE INTERACTION*, USER=INTERACTION, TRACKING
THICKNESS=*tracking_thickness*

Interfacial State

Constitutive models used to define the interfacial behavior may require the storage of solution-dependent state variables. You must allocate storage space for these variables by indicating the number of variables. There is no restriction on the number of state variables associated with a user-defined constitutive behavior for the interface.

User subroutine *UINTER* is called for points on the secondary surface at each iteration of every increment. User subroutine *VUINTER* is called in every time increment for each main-secondary view of each contact pair it affects, as discussed earlier. User subroutine *VUINTERACTION* is called in every time increment for each pair of surfaces actively interacting, as discussed earlier. Each subroutine is provided with the state of the secondary node or potential contact point at the start of the increment (the state includes stress, flux, solution-dependent state variables, temperature, and any predefined field variables) and with the increments in temperature, predefined state variables, relative position, and time.

Input File Usage: Use the following option to allocate storage space for solution-dependent state variables:

**SURFACE INTERACTION*, DEPVAR=*number_of_state_variables*

Use with the Unsymmetric Equation Solver in Abaqus/Standard

If the constitutive Jacobian matrix, $\partial \Delta \boldsymbol{\sigma} / \partial \Delta \mathbf{u}$, is not symmetric, you should invoke the unsymmetric equation solution capability in Abaqus/Standard (see *Defining an Analysis*).

Input File Usage: **SURFACE INTERACTION*, USER, UNSYMM

Defining the Contact Status in Abaqus/Standard

In addition to defining the constitutive behavior, in Abaqus/Standard you may also update the flags `LOPENCLOSE`, `LSTATE`, and `LSDI`. The flag `LOPENCLOSE` is useful when *UINTER* is used to model standard contact between two surfaces (similar to the default hard contact in Abaqus). It should be set to 0 to indicate an open status and to 1 to indicate a closed status. At the beginning of the analysis it is set to −1 before *UINTER* is called. A change in this flag from one iteration to the next will have two consequences. It will result in output related to the change in contact status if detailed contact output has been requested to the message file (see *The Abaqus/Standard Message File*), and it will also trigger a severe discontinuity iteration. The flag `LSTATE` can be used to store the current contact status of the points on the secondary surface in non-standard situations where a simple open/close status is not appropriate. An example of such a situation is debonding, where three different states can be defined—fully bonded, partially bonded or debonding, and fully debonded. You can assign an integer to each of these states and set `LSTATE` accordingly. At the beginning of the analysis `LSTATE` is set to −1 before *UINTER* is called. When this flag is used and it changes from one iteration to the next, you can output messages to the message file (unit 7) related to such a change in state directly from user subroutine *UINTER*. The flag `LPRINT` is provided to allow you to output messages related to change in contact status only when you request detailed contact output to the message file. In such a situation the `LSDI` flag may be set to 1 to trigger a severe discontinuity iteration (this issue is discussed in detail later).

An example of a situation where both the flags `LOPENCLOSE` and `LSTATE` can be used arises in the modeling of debonding between two surfaces. When the surface is in a state of transition from bonded to debonded, the flag `LSTATE` may be used, while the flag `LOPENCLOSE` may be left to its original value of −1. However, once complete debonding has taken place, the contact between the two surfaces may be modeled using standard hard contact. In that situation the `LSTATE` flag may be set to −1, and the `LOPENCLOSE` flag used. Any time one of these two flags is set to −1, Abaqus/Standard assumes that it is not being used. A change of these flags from some other value to −1 does not result in contact-status related output or severe discontinuity iterations. Similarly,

a change of these flags from −1 to some other value will not result in contact-status related output or severe discontinuity iterations.

If these flags are not used, there will be no output related to change in contact status unless you decide to output messages that are not based on these flags directly from *UINTER*.

Severe Discontinuity Iterations in Abaqus/Standard

Abaqus/Standard classifies iterations in which the contact state at the end of the iteration is different from the state assumed for that iteration as severe discontinuity iterations. The treatment of severe discontinuity iterations by Abaqus/Standard is discussed in *Severe Discontinuities in Abaqus/Standard*. When you define the interfacial constitutive behavior through user subroutine *UINTER* and do not use the `LOPENCLOSE` flag, it is your responsibility to provide Abaqus/Standard with input on how an iteration should be treated. The flag `LSDI` is provided in user subroutine *UINTER* for this purpose. It is set to 0 before each call to *UINTER*; you should set it to 1 to treat the current iteration as a severe discontinuity iteration. If the `LOPENCLOSE` flag is used, the value of this flag alone determines whether a severe discontinuity iteration is necessary or not, and the `LSDI` flag is ignored.

Use with Contact in Abaqus/Explicit

The penalty contact algorithm must be used with user subroutines *VUINTER* and *VUINTERACTION*; see *Penalty Contact Algorithm*.

When *VUINTER* is used and balanced main-secondary contact is specified (that is, the contact pair weighting factor is not equal to 0.0 or 1.0), *VUINTER* will be called for each surface in the contact pair that can act as a secondary surface. The forces and fluxes defined in *VUINTER* will be multiplied by the weight value for the main-secondary view before they are applied.

Effects on Solution Time in Abaqus/Explicit

Abaqus/Explicit accounts for the contact stiffness and conductance in the stable time increment calculation. Specifying stresses and fluxes in the user subroutine that correspond to large contact stiffness (e.g., large slope of contact pressure versus penetration) and large contact conductance will cause a significant drop in the stable time increment and, therefore, an increase in the solution time. Tangent stiffnesses and conductances are determined by Abaqus/Explicit using a finite difference method. User subroutine *VUINTER* is called three times per increment for each main-secondary view of each two-dimensional contact pair that references it and four times per increment for each three-dimensional contact pair that references it. User subroutine *VUINTERACTION* is called four times per increment for each active surface interaction that references it. The user subroutines are called once with the actual configuration and subsequently with perturbed configurations based on displacement perturbations in the normal direction, the $\mathbf{t_1}$ tangential direction, and, in three-dimensional cases, the $\mathbf{t_2}$ tangential direction, respectively (see the local coordinate system discussion in *VUINTER* and *VUINTERACTION* for an explanation of how the $\mathbf{t_1}$ and $\mathbf{t_2}$ directions are defined). For example, each component of contact stiffness is computed as a difference in contact stress divided by a difference in relative position. You do not have access to the computed values of contact stiffness and conductance, but you can control the constitutive behavior of the model. Estimated default penalty stiffness (and conductance) values are provided to the user subroutines for comparison purposes. Contact stiffnesses or conductances that exceed the default penalty values can significantly reduce the time increment size. The default penalty stiffnesses and conductances are based on an assumption that all secondary nodes are in contact. In the case of *VUINTER*, if only a fraction of the secondary nodes are in contact, higher penalties than are reported in *VUINTER* would be assigned in some cases with the default penalty algorithm.

Any changes to state variables are ignored for the perturbation calls.

In the case of *VUINTER* there can be significant additional CPU expense associated with contact tracking. Since the contact state is unknown on entry to *VUINTER*, all nodes on the secondary surface must be tracked in every increment. This can increase the cost of an analysis significantly compared to the contact models in Abaqus/Explicit if a large proportion of the secondary nodes are not involved in contact.

In the case of *VUINTERACTION* there can be significant additional CPU expense associated with contact tracking only if the tracking thickness is large compared to the element facet size on contacting surfaces.

Use with Other Subroutines

Any other user subroutine that does not deal with constitutive behavior across an interface can be used in conjunction with *UINTER*, *VUINTER*, or *VUINTERACTION*.

For example, user subroutines *UMAT* and *UMATHT* can be used in conjunction with *UINTER* to define the constitutive mechanical and thermal behaviors of the material underlying the contact surfaces. User subroutine *VUMAT* and *VUMATHT* can be used in conjunction with *VUINTER* to define the constitutive mechanical and thermal behaviors of the material underlying the contact surfaces. However, user subroutines *FRIC*, *GAPCON*, and *GAPELECTR*—available in Abaqus/Standard for defining mechanical, thermal, and electrical interactions between surfaces—can be used in conjunction with *UINTER* only if they are referenced on separate surface interactions. The same restriction applies to user subroutine *VFRIC* used in conjunction with *VUINTER* and to user subroutines *VFRICTION* or *VFRIC_COEF* used in conjunction with *VUINTERACTION*.

Use with Contact Controls

In Abaqus/Standard contact controls will not have any effect when used at an interface whose constitutive behavior is defined through user subroutine *UINTER*.

In Abaqus/Explicit contact controls can be specified for a contact pair referencing a user-defined surface interaction. In the case of user subroutine *VUINTERACTION* the default penalty stiffness argument includes any scale factor specified; whereas with user subroutine *VUINTER* the scale factor is ignored.

Output

Most of the standard output variables that are normally available in an analysis involving contact are available with this capability.

Output for `UINTER`

The variables COPEN and CSLIP represent the relative positions normal and tangential to the interface, respectively. The surface-based thermal interaction variable, SFDR, contains the heat flux due to the total energy dissipated due to friction, and not some fraction of it. This is unlike using the built-in capability in Abaqus/Standard, where SFDR may contain the heat flux due to only a fraction of the total frictional dissipation, depending on the specified fraction of the dissipated energy that is converted into heat. In addition, the surface-based thermal interaction variable WEIGHT, which represents the weighting factor for heat flux (generated by frictional sliding) distribution between the surfaces, is not available with this capability.

Additional user-defined output variables can be defined for *UINTER* by using the solution-dependent state variables (SDV).

Output for `VUINTER` and `VUINTERACTION`

All contact output variables in Abaqus/Explicit will be available except output for spot welds (BONDSTAT and BONDLOAD).

The following user subroutine variables will contribute to the associated total energy variables: the variable `sed` will contribute to the energy output variable ALLSE; `sfd` will contribute to ALLFD; `scd` will contribute to ALLCD; `spd` will contribute to ALLPD; and `svd` will contribute to ALLVD.

If SFDR is requested, `sfd`, `scd`, `spd`, and `svd` will also be used to calculate the heat generated at the interface (for output purposes only; the generated heat will not be applied to the model). The default values of the fraction of mechanical energy converted into heat and the weighting factor for the distribution of heat between the two surfaces (1.0 and 0.5, respectively) are used.

User-defined, solution-dependent state variables associated with the user subroutine cannot be output to the output database (`.odb`) file or results (`.fil`) file.

Interaction of Debonded Surfaces

Products: Abaqus/Standard

References:

- *Contact Pressure-Overclosure Relationships*
- *Frictional Behavior*
- *Thermal Contact Properties*
- *Pore Fluid Contact Properties*
- **DEBOND*
- **FRACTURE CRITERION*

Overview

This section outlines briefly how initially bonded surfaces may interact once they have started to debond. Details on defining a crack propagation analysis can be found in *Crack Propagation Analysis*.

When two initially bonded surfaces start to debond:

- the debonded secondary surface nodes are released and can move freely;
- the tractions acting on the secondary surface nodes at the instant of debonding are ramped down to zero using a user-supplied amplitude curve; and
- the contact property models assigned to the contact pair formed by the two surfaces start to govern the interaction of the surfaces.

Frictional Interactions of Debonding Surfaces

Once the surfaces start to debond, the friction model assigned to the surfaces will govern the tangential motion of the debonded secondary nodes. Friction generates forces tangential to the interface when the surfaces are closed. The frictional forces are independent of the debonding tractions that Abaqus/Standard applies and ramp off once a secondary node debonds; the debonding tractions have no influence on the frictional behavior of a surface.

Interaction Models for Behavior Normal to the Debonding Surfaces

The crack propagation capability in Abaqus/Standard was designed for use in classical fracture mechanics problems. It is intended that the capability be used with the default “hard” contact pressure-clearance model. Abaqus/Standard will prevent the use of one of the nondefault pressure-clearance models when the surfaces can debond.

Thermal Interaction of Bonded and Debonding Surfaces

Crack propagation simulations can be performed as coupled temperature-displacement analyses in Abaqus/Standard. While bonded, the surfaces are treated as having complete continuity of the temperature field across the interface. Once the surfaces start to debond, the thermal contact property models assigned to the surfaces will govern the thermal interactions across the debonded portion of the interface.

Pore Fluid Interaction of Bonded and Debonding Surfaces

Crack propagation simulations can be performed in coupled pore pressure-displacement analyses. Whether the surfaces are bonded or are debonding, they are treated as having complete continuity of the pore pressure field across the interface.

Contact Cohesive Behavior

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Progressive Damage and Failure*
- *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*
- *About Contact Pairs in Abaqus/Standard*
- *About General Contact in Abaqus/Explicit*
- *About Mechanical Contact Properties*
- *Crack Propagation Analysis*
- **COHESIVE BEHAVIOR*
- **SURFACE INTERACTION*
- **DAMAGE INITIATION*
- **DAMAGE EVOLUTION*
- **DAMAGE STABILIZATION*
- **FRACTURE CRITERION*
- *Specifying cohesive behavior properties for mechanical contact property options*
- *Specifying cohesive damage properties for mechanical contact property options*

Overview

The capability described in this section can be used to model a bonded interface, with or without the possibility of damage and failure of the bond, and to model regular contact behavior where the interface is not bonded.
This capability has similarities to other features that could be considered for a bonded interface, including cohesive elements (see *About Cohesive Elements*). Cohesive contact behavior is typically easier to define than modeling the interface using cohesive elements and allows simulation of a wider range of cohesive interactions, such as two "sticky" surfaces coming into contact during an analysis.

Contact cohesive behavior is primarily intended for situations in which the interface thickness is negligibly small. If the interface adhesive layer has a finite thickness and macroscopic properties (such as stiffness and strength) of the adhesive material are available, it might be more appropriate to model the response using conventional cohesive elements (see *Defining the Constitutive Response of Cohesive Elements Using a Continuum Approach*).

In Abaqus/Explicit the surface-based cohesive behavior framework can also be used to model crack propagation in initially partially bonded surfaces via linear elastic fracture mechanics principles (LEFM) as implemented using the Virtual Crack Closure Technique (VCCT).

Contact cohesive behavior:

- is defined as a surface interaction property;
- can be used to model the delamination at interfaces directly in terms of traction versus separation;
- can be used to model "sticky" contact (that is, surfaces or parts of surfaces that are not initially in contact might bond on coming into contact; subsequently the bond might damage and fail);
- can be restricted to surface regions that are initially in contact;
- allows specification of cohesive data such as the fracture energy as a function of the ratio of normal to shear displacements (mode mix) at the interface;
- assumes a linear elastic traction-separation law prior to damage;

- assumes that failure of the cohesive bond is characterized by progressive degradation of the cohesive stiffness, which is driven by a damage process (in Abaqus/Explicit brittle fracture can also be modeled using a VCCT fracture criterion);
- allows specification of postfailure cohesive behavior if failed nodes re-enter contact;
- is implemented within the general contact algorithmic framework in Abaqus/Standard and Abaqus/Explicit and within the contact pair framework in Abaqus/Standard;
- is enforced with the surface-to-surface, edge-to-surface, edge-to-edge, and vertex-to-surface contact formulations for general contact in Abaqus/Standard;
- is enforced for node-to-face contact interactions and for edge-to-edge contact interactions for edges associated with circular beams in Abaqus/Explicit;
- is not available for edge-to-edge contact interactions for edges not associated with circular beams or for node-to-analytical rigid surface interactions in Abaqus/Explicit;
- is enforced for the node-to-surface contact formulation for contact pairs in Abaqus/Standard;
- is not available for the finite-sliding, surface-to-surface contact formulation for contact pairs in Abaqus/Standard;
- can be used as an alternative to “rough friction” surface interactions, the “no separation” contact relationship, or a combined “no separation and rough friction” behavior within the general contact framework;
- is an alternative way to tie surfaces; and
- cannot be used in a coupled Eulerian-Lagrangian analysis in Abaqus/Explicit.

Common Usages of Cohesive Contact Behavior

Cohesive contact can be used in a variety of workflows. Cohesive contact behavior often is one of many possible approaches to modeling interface behavior. Common usages of cohesive contact include:

- Modeling a permanently bonded interface.
- Modeling a bonded interface in which the bond might damage and fail.
- Approximating interface behavior in a simplified form while a model is being built (and other aspects of the model are being refined).

These usages are discuss in more detail below.

Modeling a Permanently Bonded Interface

In it simplest form, cohesive contact can be used as an alternative to surface-based tie constraints (which are discussed in *Mesh Tie Constraints*) or other modeling methods. There is no need to specify stiffness or damage properties of the contact cohesive behavior in this case; you can allow Abaqus to assign default interfacial stiffness components. Bonded regions remain bonded throughout a simulation if cohesive damage characteristics are not specified. Unlike surface-based tie constraints, cohesive contact will not constrain rotational degrees of freedom.

Modeling a permanently bonded interface as a type of contact behavior rather than as a surface-based tie constraint has the following advantages:

- Enables contact output variables to be used to evaluate interface stresses and other quantities.
- Enables numerical softening to be introduced in the constraint enforcement, which avoids the potential for numerical issues associated with overconstraints where different types of strictly enforced "hard" constraints overlap.
- Optionally, allows a specific interface stiffness representative of physical behavior to be specified.

Permanent cohesive bonds with default cohesive stiffness or user-specified cohesive stiffness at least as stiff as the default cohesive stiffness have the following characteristics for general contact in Abaqus/Standard:

- No regular contact constraints act in parallel to cohesive contact constraints: Conditions for regular contact constraints acting in parallel to cohesive contact constraints are discussed in *Interaction between Cohesive Properties and Regular Contact Properties*. However, those conditions are not relevant to stiff, permanent cohesive constraints, and regular contact constraints are avoided in these cases to improve convergence behavior and performance.
- Details of the cohesive contact formulation are enhanced for stiff, permanent cohesive bonds. Results for stiff cohesive bonds might differ, even prior to cohesive damage, depending on whether or not a cohesive damage evolution model is specified.

Modeling a Bonded Interface That May Fail

Specifying a damage model for the contact cohesive behavior allows for modeling of a bonded interface that might fail as a result of the loading. This modeling approach is an alternative to using cohesive elements or other element types that directly discretize the cohesive material for the simulation. Comparisons of cohesive-contact versus cohesive-element approaches are discussed below in *High-Level Comparison of Cohesive-Element and Cohesive-Contact Approaches*.

Approximating and Modifying Interface Behavior While a Finite Element Model Is Built

Using different interface modeling strategies across different stages of building and refining a finite element model is sometimes a good strategy for improving your efficiency. For example, during an initial stage of a model build, you might choose to model interfaces as permanently bonded to enable more focus on noninterface modeling details. You can switch to more physically representative interface behavior (such as regular contact or bonded contact with the possibility of damage and failure) in later stages of the model build. The later stages often require more care to avoid unconstrained rigid body modes and other types of static instabilities.

Analysts sometimes use surface-based tie constraints (*Mesh Tie Constraints*) in early stages of building a model and then switch to contact specifications as the model becomes more mature. An alternative is to specify cohesive contact behavior with a permanently bonded interface and default stiffness in the early stages, and then reassign a more realistic contact behavior as the model becomes more mature. This alternative of reassigning the contact behavior as the model matures, rather than switching from a constraint option to a contact option during the model evolution, might result in greater consistency across different stages of the model build.

High-Level Comparison of Cohesive-Element and Cohesive-Contact Approaches

Figure 1 provides a high-level comparison of the cohesive-element and cohesive-contact modeling approaches. Both of these approaches are viable for many modeling situations.

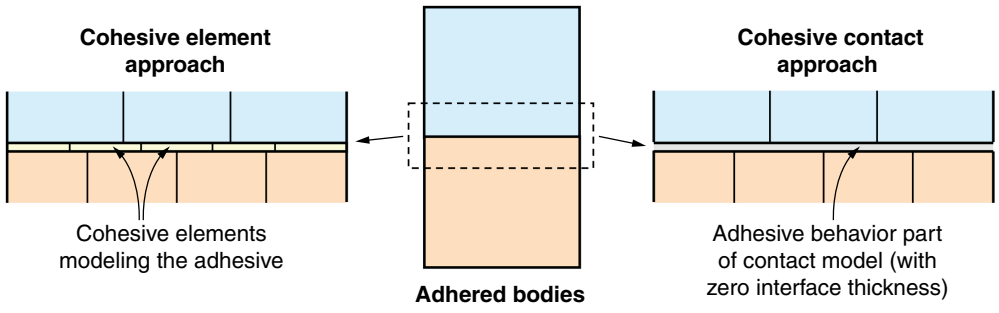


Figure 1: Schematic comparison of cohesive element and cohesive contact approaches.

The formulas and laws that govern cohesive constitutive behavior are very similar for cohesive contact and cohesive elements. The similarities extend to the linear elastic traction-separation model, damage initiation criteria, and damage evolution laws. Constitutive behavior details for contact cohesive behavior are discussed

later in this section, starting with *Cohesive Stiffness*. Constitutive behavior details for cohesive element are discussed in *Defining the Constitutive Response of Cohesive Elements Using a Traction-Separation Description*.

It is important to recognize differences between the cohesive-contact and cohesive-element approaches, including the aspects discussed below.

No Cohesive Contact Thickness

Cohesive material thickness cannot be introduced as a characteristic for cohesive contact but can be for cohesive elements. Surface thickness can be modified (*Assigning Surface Properties*) to account for cohesive material thickness. Since thickness effects are not considered for a cohesive property, material definitions used to describe traction-separation response for cohesive elements with thickness effects might not be directly reusable for cohesive contact.

Tangential Refinement of the Interface

Constitutive calculations are evaluated for cohesive contact and cohesive elements at the following locations:

- For cohesive elements, constitutive calculations are calculated at the material points of the elements.
- For cohesive contact, constitutive calculations are calculated for individual contact constraints. The number of potential contact constraints is approximately equal to the number of nodes acting as secondary nodes.

Modeling with cohesive elements allows the possibility of different tangential mesh refinement for cohesive elements as compared to the mesh refinement of the adjacent bodies. Use of a more refined mesh for the cohesive elements might improve the resolution of spatial variations in the cohesive response, independent, to a degree, of the mesh refinement of the adjacent bodies. The cohesive element example in *Figure 1* shows a slightly more refined mesh for the cohesive elements than the adjacent bodies.

For the cohesive contact modeling approach, cohesive calculations are computed at contact constraint locations, which are primarily associated with secondary nodes. The more refined surface of an interaction typically acts as the secondary surface. Therefore, the resolution of spatial variations in the cohesive response is usually primarily associated with whichever adjacent body has the more refined surface.

Interaction between Cohesive Properties and Regular Contact Properties

"Regular" contact behavior is automatically in effect if the cohesive contact bond becomes fully damaged. Cohesive elements do not have an analogous behavior in this regard unless contact is defined between surfaces of the adhered parts in addition to having cohesive elements defined between the adhered parts. A surface interaction property definition containing cohesive specifications will also include noncohesive, mechanical contact specifications, such as discussed in *Contact Pressure-Overclosure Relationships* and *Frictional Behavior*.

"Regular" contact-behavior aspects are sometimes partially in effect even before the cohesive has failed, as described below:

- Normal-direction behavior: The noncohesive contact pressure-overclosure relationship (see *Contact Pressure-Overclosure Relationships*) is in effect while the contact pressure is positive, regardless of whether cohesive behavior is specified and the amount of cohesive damage accumulated except for stiff, permanent cohesive behavior. No regular contact constraints act in parallel to cohesive contact constraints for stiff, permanent cohesive behavior.
- Tangential behavior: If cohesive bonding at a particular interface location is active and undamaged, the resistance to tangential motion is governed by the cohesive behavior only. Once cohesive damage starts to accumulate at a particular location of the interface, the interface shear stress has contributions from the cohesive model and the friction model. The contribution from the friction model is weighted by the scalar damage variable of the cohesive behavior (see *Damage Evolution*). When the cohesive bond is fully damaged (failed), the only contribution to the interface shear stress is from the friction model.

Interface Versus Element Quantities

The table below compares how various simulation operations associated with cohesive modeling can be performed with the cohesive contact and cohesive element modeling approaches.

Simulation operation	Cohesive contact	Cohesive elements
Defining where a cohesive region is located	Interaction property assignment (based on surface pairings)	Including cohesive elements (and nodes) in the model
Defining cohesive damage model and other aspects of cohesive constitutive behavior	Interaction property specification	Material property specification
Studying results for stretching and shearing of a cohesive material	Contact opening and sliding distance output	Element strain output
Studying results for stresses within a cohesive material	Contact stresses output for normal and tangent directions	Element stress output

Specifying Cohesive Interface "Material" Behavior within a Surface Interaction Property Definition

Cohesive interface "material" behavior is defined as part of a surface interaction property. Surface interaction properties are assigned to contact interactions as discussed in *Defining the Contact Property Model*. Cohesive interface behavior includes stiffness characteristics associated with the bonded interface and characteristics governing any cohesive damage. Bonded-interface stiffness characteristics are assigned by default if these stiffness characteristics are not specified explicitly. The magnitudes of these default stiffness characteristics are similar to the magnitude of the default contact penalty stiffness. A damage model is not included in the cohesive material behavior unless damage characteristics are specified explicitly as part of the damage behavior definition.

Input File Usage: Use the following options to define cohesive behavior as part of a surface interaction definition:

**SURFACE INTERACTION*
**COHESIVE BEHAVIOR*

Abaqus/CAE Usage: Use the following option to define cohesive behavior as part of a surface interaction definition:

Interaction module: contact property editor: **Mechanical->Cohesive Behavior**

Initial Cohesive Contact State

The initial contact status as a function of position along a cohesive contact interface can fundamentally affect simulation results. Consider the example shown in *Figure 2*. The intent for this example is that the block is initially touching the wall with the cohesive status initialized to bonded. However, a small, unintended initial gap exists between the block and the wall in the initial configuration, so the contact status is initialized to "opened" or "inactive," and the cohesive status is initialized to unbonded by default. If there is no initial cohesive bonding in this example, the applied force will push the block away from the wall or perhaps, in a static analysis, a numerical issue will be reported by Abaqus/Standard due to unconstrained rigid-body motion of the block. User controls associated with the initial contact status (see *Contact Initialization for General Contact in Abaqus/Standard*, *Contact Initialization for Contact Pairs in Abaqus/Standard*, and *Contact Initialization for General Contact in Abaqus/Explicit*) can be used to ensure that the contact status will be properly initialized over various regions of an interface, such that interface stresses associated with cohesive contact will counter the applied force. Most user controls associated with the initial contact status are not specific to cohesive contact behavior.

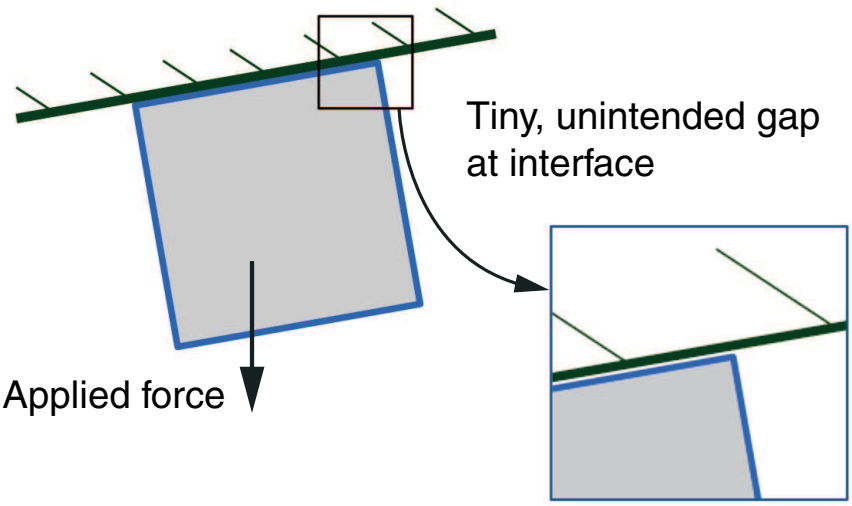


Figure 2: Block intended to be in cohesive contact with a rigid surface.

Consider the example shown in *Figure 3*, in which the cohesive status is intended to be initialized to bonded over much of the interface but should be initialized to unbonded over a specific portion of the interface. The required initialization can be achieved by assigning zero initial clearance to the portion of the interface that should be initially bonded and very small positive initial clearance to the portion of the interface that should not be initially bonded, such as shown in the Abaqus/Standard input file example below.

```
*CONTACT INITIALIZATION DATA, NAME=TOUCHING, INITIAL CLEARANCE=0.0, SEARCH
 ABOVE=0.1
*CONTACT INITIALIZATION DATA, NAME=SMALL_GAP, INITIAL CLEARANCE=1.E-7,
SEARCH ABOVE=0.1
…
*CONTACT
…
*CONTACT INITIALIZATION ASSIGNMENT
BLOCK_SURF_BONDED_REGION, RIGID_SURF, TOUCHING
BLOCK_SURF_UNBONDED_REGION, RIGID_SURF, SMALL_GAP
```

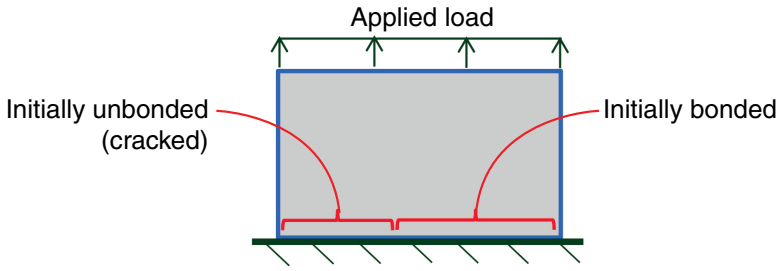


Figure 3: Block intended to be initially bonded over a portion of an interface with a rigid surface.

Limiting Cohesive Bonding to Original Contact Constraints

The most common usage of cohesive contact is for situations in which cohesive bonds exist at the beginning of a simulation. By default, Abaqus limits cohesive bonds to those that exist at the beginning of a simulation.

Input File Usage: Use either of the following options to limit cohesive behavior to original contact constraints:

**COHESIVE BEHAVIOR*, COHERE=ORIGINAL CONTACTS
**COHESIVE BEHAVIOR*, ELIGIBILITY=ORIGINAL CONTACTS

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Cohesive Behavior**: **Only secondary nodes initially in contact**

Limiting Cohesive Bonding to Subset of Original Contact Constraints

For contact pairs in Abaqus/Standard you can specify as part of the cohesive behavior definition that only a subset of initially active contact constraints should have cohesive bonds. Initial strain-free adjustments to positions of secondary nodes will be made, if necessary, to ensure they are initially in contact with the main surface. Similar behavior can be achieved with general contact by selectively assigning initialization controls to control which regions of the interface are initially in contact and limiting cohesive behavior to initially active contact constraints (see *Initial Cohesive Contact State*).

Input File Usage: Use both of the following options:

**INITIAL CONDITIONS*, TYPE=CONTACT
**COHESIVE BEHAVIOR*, ELIGIBILITY=SPECIFIED CONTACTS

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Cohesive Behavior**: **Specify the bonding node set in the surface-to-surface (Standard) interaction**

Interaction module: interaction editor: **Bonding** tabbed page: **Limit bonding to secondary nodes in subset**

Cohesive Rebonding upon Repeated Contact

In some situations it is desirable to allow cohesive rebonding each time contact is established, even for secondary nodes previously involved in cohesive contact that have fully damaged and debonded. For such situations, you can indicate that cohesive rebonding can repeatedly occur at the same interface location.

Input File Usage: Use either of the following options to allow rebonding each time contact is reestablished:

**COHESIVE BEHAVIOR*, COHERE=REPEATED CONTACTS
**COHESIVE BEHAVIOR*, ELIGIBILITY=CURRENT CONTACTS, REPEATED CONTACTS

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Cohesive Behavior**: **Allow cohesive behavior during repeated post-failure contacts**

Cohesive Rebonding upon Repeated Contact Limited to Locations of Initial Cohesive Bonds

General contact in Abaqus/Explicit and contact pairs in Abaqus/Standard allow cohesive bonding to be limited to originally active contact constraints with only these secondary nodes to be eligible to rebond upon subsequent contact, and contact pairs in Abaqus/Standard allow this behavior for a subset of initially active contact constraints.

Limiting Cohesive Bonding to First Contact Constraints

It is sometimes desirable to establish cohesive bonds for initial contact constraints plus the first time an initially not-in-contact region comes into contact during a simulation. General contact in Abaqus/Explicit and contact pairs in Abaqus/Standard optionally support each secondary node associated with interactions that are assigned a cohesive property to become bonded once (either initially or during a simulation). Simulation results with this option can be highly sensitive to the assignment of secondary and main roles since the check for prior cohesive bonds at a location is done only for nodes acting as secondary nodes. General contact in Abaqus/Standard allows cohesive behavior to be limited to initial contact constraints (see *Limiting Cohesive Bonding to Original Contact Constraints*) and allows cohesive behavior for all new contact constraints (see *Cohesive Rebonding upon Repeated Contact*) but does not support limiting cohesive behavior to first contact constraints.

When cohesive contact behavior applies to contact that develops after the start of the simulation, cohesive effects are activated one increment after the contact constraint becomes active.

Input File Usage: Use either of the following options to limit cohesive bonding to first contact constraints:

**COHESIVE BEHAVIOR*, COHERE=FIRST CONTACTS
**COHESIVE BEHAVIOR*, ELIGIBILITY=CURRENT CONTACTS

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Cohesive Behavior**: **Any secondary nodes experiencing contact**

Main and Secondary Roles and Contact Formulations Associated with Cohesive Interactions

Interactions assigned a cohesive surface interaction property are modeled with pure main-secondary roles in contact formulations other than edge-to-edge contact. Both edges involved in edge-to-edge contact have equal roles. The main and secondary roles for cohesive contact with other formulations are established as follows:

- General contact in Abaqus/Standard: main and secondary roles for interactions associated with cohesive behavior are the same as for other types of contact behavior (see *Main and Secondary Surface Roles of a Contact Formulation*).
- General contact in Abaqus/Explicit: main and secondary roles for interactions associated with cohesive behavior follow the convention that the first surface specified in a contact property assignment involving cohesive behavior is treated as a secondary surface and the second surface is treated as its corresponding main surface..
- Contact pairs in Abaqus/Standard: main and secondary roles are defined by the usual conventions associated with defining a contact pair.

Cohesive Contact Normal Direction in Abaqus/Explicit

By default, the contact normal direction for Abaqus/Explicit contact with cohesive behavior is based solely on the main surface. Alternatively, you can specify that the main and secondary surfaces influence the evolution of the cohesive contact normal direction equally, which is more consistent with the behavior obtained using cohesive elements. In both cases the contact normal direction at first cohesion is determined solely by the main surface. Most cohesive bonds have surface normal directions that initially are nearly aligned but opposing one another.

The non-default option averages the effects of changes in the main and secondary surface normal directions when it updates the cohesive contact normal direction. This approach leads to more intuitive evolution of the cohesive contact normal direction in peeling tests involving ductile cohesive bonds, as shown in *Figure 4*. This image depicts a highly ductile, initially flat cohesive interface that remains actively bonded. Here, the cohesive

strain is aligned with the cohesive contact normal direction if the cohesive contact normal direction evolves with the consideration of changes in the main and secondary surface normal directions. However, if you use the default setting, Abaqus/Explicit considers this cohesive strain to have normal and shear components.

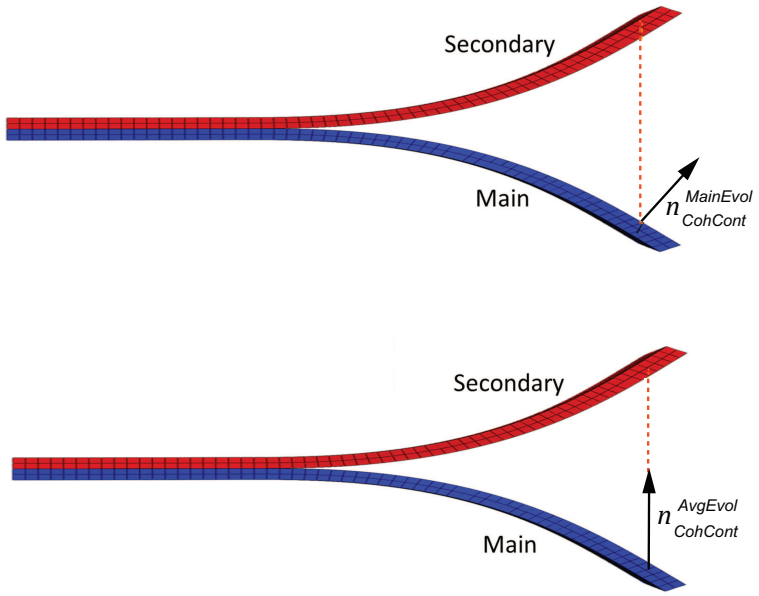


Figure 4: Contact normal directions in peeling tests involving ductile bonds. Top: Cohesive contact normal direction evolving with the main surface. Bottom: Cohesive contact normal direction evolving with the average of the main and secondary surfaces.

Figure 5 compares plots of the cohesive normal strain (COPEN) for cases in which the cohesive failure occurs when the cohesive strain reaches 20 mm. If you specify that the cohesive contact normal direction evolves with the consideration of changes in the main and secondary surface normal directions, COPEN reaches a value very close to 20 mm. If you specify the default behavior, where the contact normal direction is based solely on the main surface, the maximum normal cohesive strain component is 11.63 mm when the total cohesive strain magnitude reaches 20 mm (implying that the cohesive shear strain magnitude is 16.27 mm).

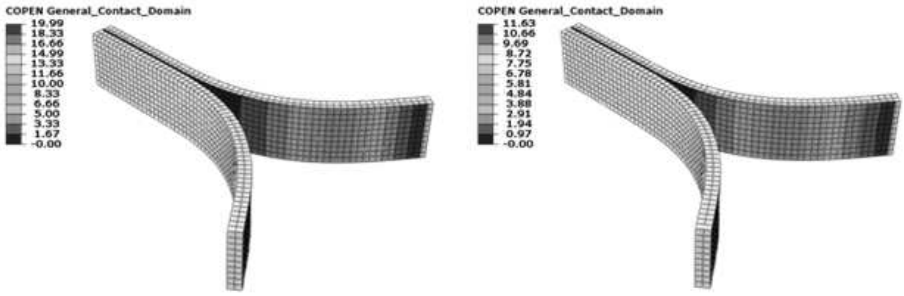


Figure 5: Cohesive normal strain (COPEN) for a peeling test with maximum cohesive strain magnitude of 20 mm using the non-default option (left) and the default option (right) for the evolution of the cohesive contact normal direction.

Input File Usage: Use the following option to calculate the contact normal direction based on the evolution of the main surface normal direction (default):

**CONTACT CONTROLS ASSIGNMENT*, NORMAL EVOLUTION=MAIN

Use the following option to calculate the contact normal direction based on the evolution of both the main and secondary surface normal directions:

**CONTACT CONTROLS ASSIGNMENT*, NORMAL EVOLUTION=AVERAGE

Cohesive Stiffness

You can choose to have the default cohesive stiffness remain in effect, scale the default cohesive stiffness, or directly specify cohesive stiffness with or without coupling among normal and tangential directions.

Default Cohesive Stiffness

By default, Abaqus assigns the contact penalty stiffness to normal and the tangential cohesive stiffness components to resist all relative motion at a cohesive contact interface. Using the default cohesive stiffness is convenient, for example, when cohesive contact represents a thin, stiff adhesive. The default penalty stiffness is chosen to be stiff without having adverse effects on numerics. If you scale the contact penalty stiffness (see *About General Contact in Abaqus/Explicit*), the default cohesive stiffness components will be likewise scaled. For example, if relative motion at a cohesive contact bond is larger than desired with the default cohesive stiffness in effect, you can consider increasing the penalty stiffness by a factor of three or ten.

The nominal traction stress vector, $\mathbf{t}$, consists of three components (two components in two-dimensional problems): t_n, t_s, and (in three-dimensional problems) t_t, which represent the normal (along the local 3-direction in three dimensions and along the local 2-direction in two dimensions) and the two shear tractions (along the local 1- and 2-directions in three dimensions and along the local 1-direction in two dimensions), respectively. The corresponding relative motion components are denoted by δ_n, δ_s, and δ_t.

The default relationship between cohesive traction stress components and relative motion components is

$$\boldsymbol{t} = \left\{ \begin{array}{c} t_n \\ t_s \\ t_t \end{array} \right\} = k_{Pnlty} \left\{ \begin{array}{c} \delta_n \\ \delta_s \\ \delta_t \end{array} \right\} = k_{Pnlty} \boldsymbol{\delta}.$$

Uncoupled Cohesive Stiffness

You can specify the normal direction cohesive stiffness, K_{nn}, and the tangential stiffness components, K_{ss}, and K_{tt}, with optional dependence on temperature and field variables. For this case of uncoupled cohesive stiffness, relative displacement in the normal direction does not give rise to cohesive shear forces and shear tractions, and relative tangential displacement does not give rise to cohesive normal forces and normal tractions. Points of interaction on respective surfaces are established when each cohesive contact constraint is formed and do not evolve while the cohesive contact constraint is active. K_{nn}, K_{ss}, and K_{tt} should be positive (nonzero) values:

$$\mathbf{t} = \left\{ \begin{array}{c} t_n \\ t_s \\ t_t \end{array} \right\} = \left[\begin{array}{ccc} K_{nn} & 0 & 0 \\ 0 & K_{ss} & 0 \\ 0 & 0 & K_{tt} \end{array} \right] \left\{ \begin{array}{c} \delta_n \\ \delta_s \\ \delta_t \end{array} \right\} = \mathbf{K}\boldsymbol{\delta}.$$

Input File Usage: **COHESIVE BEHAVIOR*, TYPE=UNCOUPLED (default)

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Cohesive Behavior**: **Specify stiffness coefficients**: **Uncoupled**

Coupled Cohesive Stiffness

In its full generality, the cohesive stiffness matrix provides fully coupled behavior between all components of the traction vector and relative motion vector. The specified cohesive stiffness components, K_{nn}, K_{ss}, K_{tt}, K_{ns}, K_{nt}, and K_{st}, can depend on temperature or field variables. All terms in the matrix must be defined for coupled traction-separation behavior, and K_{nn}, K_{ss}, and K_{tt} should be positive (nonzero) values:

$$\mathbf{t} = \begin{Bmatrix} t_n \\ t_s \\ t_t \end{Bmatrix} = \begin{bmatrix} K_{nn} & K_{ns} & K_{nt} \\ K_{ns} & K_{ss} & K_{st} \\ K_{nt} & K_{st} & K_{tt} \end{bmatrix} \begin{Bmatrix} \delta_n \\ \delta_s \\ \delta_t \end{Bmatrix} = \mathbf{K}\boldsymbol{\delta}.$$

Input File Usage: **COHESIVE BEHAVIOR*, TYPE=COUPLED

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Cohesive Behavior**: **Specify stiffness coefficients**: **Coupled**

Damage Modeling

Damage modeling allows you to simulate the degradation and eventual failure of the bond between two cohesive surfaces. The failure mechanism consists of two ingredients: a damage initiation criterion and a damage evolution law. The initial response is assumed to be linear as discussed above. However, once a damage initiation criterion is met, damage can occur according to a user-defined damage evolution law. *Figure 6* shows a typical traction-separation response with a failure mechanism. If the damage initiation criterion is specified without a corresponding damage evolution model, Abaqus evaluates the damage initiation criterion for output purposes only; there is no effect on the response of the cohesive surfaces (i.e., no damage will occur). Cohesive surfaces do not undergo damage under pure compression.

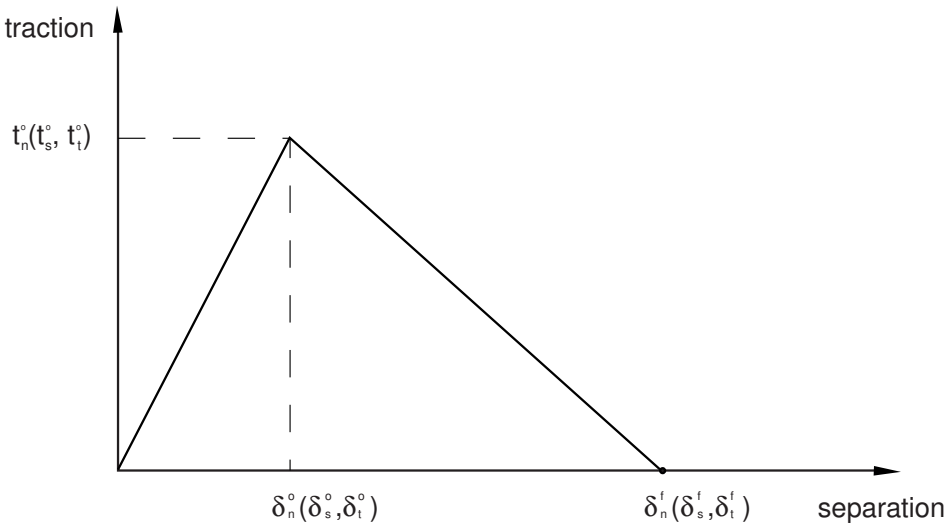


Figure 6: Typical traction-separation response.

Damage of the traction-separation response for cohesive surfaces is defined within the same general framework used for conventional materials (see *About Progressive Damage and Failure*), except the damage behavior is specified as part of the interaction properties for the surfaces. Multiple damage response mechanisms are not

available for cohesive surfaces: cohesive surfaces can have only one damage initiation criterion and only one damage evolution law.

Input File Usage: Use the following options to define damage initiation and damage evolution for cohesive surfaces:

**SURFACE INTERACTION*, NAME=*name*
**COHESIVE BEHAVIOR*
**DAMAGE INITIATION*
**DAMAGE EVOLUTION*

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical**->**Damage**: **Damage Initiation** and **Damage Evolution** tabbed pages

Damage Initiation

Damage initiation refers to the beginning of degradation of the cohesive response at a contact point. The process of degradation begins when the contact stresses and/or contact separations satisfy certain damage initiation criteria that you specify. Several damage initiation criteria are available and are discussed below.

Each damage initiation criterion also has an output variable associated with it to indicate whether the criterion is met. A value of 1 or higher indicates that the initiation criterion has been met. Damage initiation criteria that do not have an associated evolution law affect only output. Thus, you can use these criteria to evaluate the propensity of the material to undergo damage without actually modeling the damage process (that is, without actually specifying damage evolution).

In the discussion below, t_n^o, t_s^o, and t_t^o represent the peak values of the contact stress when the separation is either purely normal to the interface or purely in the first or the second shear direction, respectively. Likewise, δ_n^o, δ_s^o, and δ_t^o represent the peak values of the contact separation, when the separation is either purely along the contact normal or purely in the first or the second shear direction, respectively. The symbol $\langle\rangle$ used in the discussion below represents the Macaulay bracket with the usual interpretation. The Macaulay brackets are used to signify that a purely compressive displacement (that is, a contact penetration) or a purely compressive stress state does not initiate damage.

Maximum Stress Criterion

Damage is assumed to initiate when the maximum contact stress ratio (as defined in the expression below) reaches a value of one. This criterion can be represented as

$$\max\left\{\frac{\langle t_n\rangle}{t_n^o}, \frac{t_s}{t_s^o}, \frac{t_t}{t_t^o}\right\} = 1.$$

Input File Usage: **DAMAGE INITIATION*, CRITERION=MAXS

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical**->**Damage**: **Initiation** tabbed page: **Criterion**: **Maximum nominal stress**

Maximum Separation Criterion

Damage is assumed to initiate when the maximum separation ratio (as defined in the expression below) reaches a value of one. This criterion can be represented as

$$\max\left\{\frac{\langle\delta_n\rangle}{\delta_n^o}, \frac{\delta_s}{\delta_s^o}, \frac{\delta_t}{\delta_t^o}\right\} = 1.$$

Input File Usage: **DAMAGE INITIATION*, CRITERION=MAXU

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damage**: **Initiation** tabbed page: **Criterion**: **Maximum separation**

Quadratic Stress Criterion

Damage is assumed to initiate when a quadratic interaction function involving the contact stress ratios (as defined in the expression below) reaches a value of one. This criterion can be represented as

$$\left\{\frac{\langle t_n\rangle}{t_n^o}\right\}^2 + \left\{\frac{t_s}{t_s^o}\right\}^2 + \left\{\frac{t_t}{t_t^o}\right\}^2 = 1.$$

Input File Usage: **DAMAGE INITIATION*, CRITERION=QUADS

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damage**: **Initiation** tabbed page: **Criterion**: **Quadratic traction**

Quadratic Separation Criterion

Damage is assumed to initiate when a quadratic interaction function involving the separation ratios (as defined in the expression below) reaches a value of one. This criterion can be represented as

$$\left\{\frac{\langle\delta_n\rangle}{\delta_n^o}\right\}^2 + \left\{\frac{\delta_s}{\delta_s^o}\right\}^2 + \left\{\frac{\delta_t}{\delta_t^o}\right\}^2 = 1.$$

Input File Usage: **DAMAGE INITIATION*, CRITERION=QUADU

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damage**: **Initiation** tabbed page: **Criterion**: **Quadratic separation**

Dependence of Damage Initiation on Effective Rate of Separation

The damage initiation criterion can depend on the effective rate of separation. For a discussion of filtering the effective rate of separation and interpolation with respect to the effective rate of separation, see *About Mechanical Contact Properties*.

Input File Usage: Use the following option to describe rate-dependent damage initiation behavior:

**DAMAGE INITIATION*, RATE DEPENDENT

Abaqus/CAE Usage: Defining rate-dependent damage initiation behavior is not supported in Abaqus/CAE.

Damage Evolution

The damage evolution law describes the rate at which the cohesive stiffness is degraded once the corresponding initiation criterion is reached. The general framework for describing the evolution of damage in bulk materials (as opposed to interfaces modeled using cohesive surfaces) is described in *Damage Evolution and Element*

Removal for Ductile Metals. Conceptually, similar ideas apply for describing damage evolution in cohesive surfaces.

A scalar damage variable, D, represents the overall damage at the contact point. It initially has a value of 0. If damage evolution is modeled, D monotonically evolves from 0 to 1 upon further loading after the initiation of damage. The contact stress components are affected by the damage according to

$$t_n = \begin{cases} (1-D)\,\bar{t}_n, & \bar{t}_n \geq 0 \\ \bar{t}_n, & \text{otherwise (no damage to compressive stiffness);} \end{cases}$$

$$t_s = (1-D)\,\bar{t}_s,$$

$$t_t = (1-D)\,\bar{t}_t,$$

where $\bar{t}_n$, $\bar{t}_s$, and $\bar{t}_t$ are the contact stress components predicted by the elastic traction-separation behavior for the current separations without damage.

To describe the evolution of damage under a combination of normal and shear separations across the interface, it is useful to introduce an effective separation (Camanho and Davila, 2002) defined as

$$\delta_m = \sqrt{\langle \delta_n \rangle^2 + \delta_s^2 + \delta_t^2}.$$

While this formula was originally applied to damage evolution in cohesive elements, it can be reinterpreted in terms of contact separations for cohesive surface behavior, as discussed above (see *High-Level Comparison of Cohesive-Element and Cohesive-Contact Approaches*).

Mixed-Mode Definition

The relative proportions of normal and shear separations at a contact point define the mode mix at the point. Abaqus uses three measures of mode mix, two that are based on energies and one that is based on tractions. You can choose one of these measures when you specify the mode dependence of the damage evolution process.

Denoting by G_n, G_s, and G_t the work done by the tractions and their conjugate separations in the normal, first, and second shear directions, respectively, and defining $G_T = G_n + G_s + G_t$, the mode mix definitions based on energies are as follows:

$$m_1 = \frac{G_n}{G_T},$$

$$m_2 = \frac{G_s}{G_T},$$

$$m_3 = \frac{G_t}{G_T}.$$

Clearly, only two of the three quantities defined above are independent. It is also useful to define the quantity $G_S = G_s + G_t$ to denote the portion of the total work done by the shear traction and the corresponding separation components. As discussed later, Abaqus requires that you specify material properties related to damage evolution as functions of $m_2 + m_3$ $(= G_S/G_T)$ (or, equivalently, $1 - m_1$) and $m_3/(m_2 + m_3)$ $(= G_t/G_S)$.

Abaqus computes the energy quantities described above either based on the current state of deformation (nonaccumulative measure of energy) or based on the deformation history (accumulative measure of energy) at an integration point. The former approach, available only in Abaqus/Standard, is useful in mixed-mode simulations where the primary energy dissipation mechanism is associated with the creation of new surfaces due to failure in the cohesive zone. Such problems are typically adequately described utilizing the methods of linear elastic fracture mechanics. The latter approach provides an alternate way of defining the mode mix and might be useful in situations where other significant dissipation mechanisms also govern the overall structural response.

The corresponding definitions of the mode mix based on traction components are given by

$$\phi_1 = \left(\frac{2}{\pi}\right) \tan^{-1}\left(\frac{\tau}{\langle t_n \rangle}\right),$$

$$\phi_2 = \left(\frac{2}{\pi}\right) \tan^{-1}\left(\frac{t_t}{t_s}\right),$$

where $\tau = \sqrt{t_s^2 + t_t^2}$ is a measure of the effective shear traction. The angular measures used in the above definition (before they are normalized by the factor $2/\pi$) are illustrated in *Figure 7*.

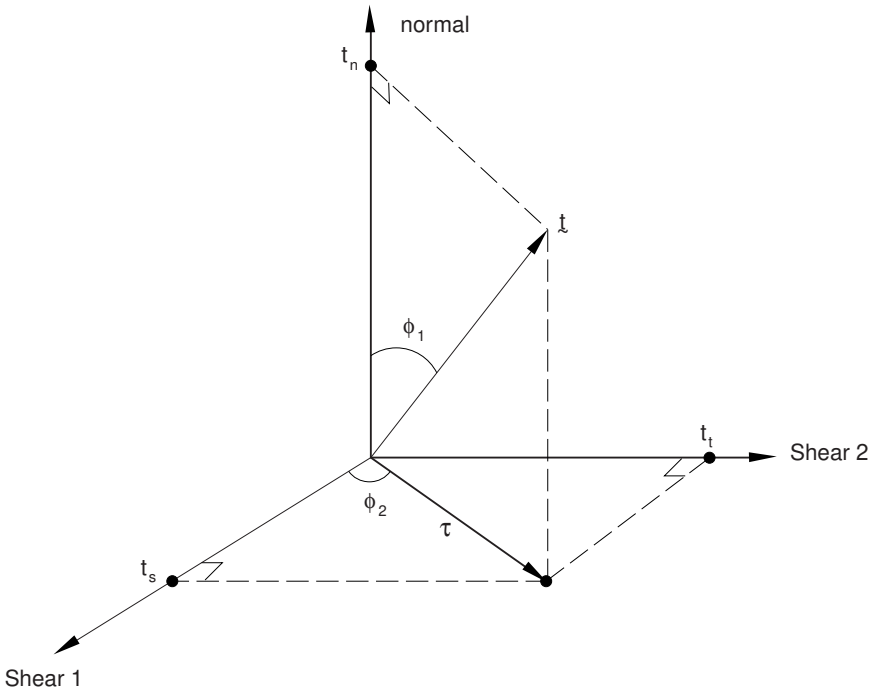


Figure 7: Mode mix measures based on traction.

Input File Usage: Use the following option to use the mode mix definition based on nonaccumulated energies (available only in Abaqus/Standard):

**DAMAGE EVOLUTION*, MODE MIX RATIO=ENERGY

Use the following option to use the mode mix definition based on accumulated energies:

**DAMAGE EVOLUTION*, MODE MIX RATIO=ACCUMULATED ENERGY

Use the following option to use the mode mix definition based on tractions:

**DAMAGE EVOLUTION*, MODE MIX RATIO=TRACTION

Abaqus/CAE Usage: Use the following option in Abaqus/Standard to use the mode mix definition based on nonaccumulated energies:

Interaction module: contact property editor: **Mechanical->Damage**: **Evolution** tabbed page: toggle on **Specify mixed-mode behavior**: **Mode mix ratio**: **Energy**

Use the following option in Abaqus/Explicit to use the mode mix definition based on accumulated energies:

Interaction module: contact property editor: **Mechanical->Damage**: **Evolution** tabbed page: toggle on **Specify mixed-mode behavior**: **Mode mix ratio**: **Energy**

Specifying a mode mix definition based on accumulated energies in Abaqus/Standard is not supported in Abaqus/CAE.

Use the following option to use the mode mix definition based on tractions:

Interaction module: contact property editor: **Mechanical->Damage**: **Evolution** tabbed page: toggle on **Specify mixed-mode behavior**: **Mode mix ratio**: **Traction**

Comparison of Mixed-Mode Definitions

The mode mix ratios defined in terms of energies and tractions can be quite different in general. The following example illustrates this point. In terms of energies a separation in the purely normal direction is one for which $G_n \neq 0$ and $G_s = G_t = 0$, irrespective of the values of the normal and the shear tractions. In particular, for coupled traction-separation behavior both the normal and shear tractions may be nonzero for a purely normal separation. For this case the definition of mode mix based on energies would indicate a purely normal separation, while the definition based on tractions would suggest a mix of both normal and shear separation.

When the mode mix is defined based on accumulated energies, an artificial path-dependence may be introduced in the mixed-mode behavior that may not be consistent, for example, with predictions that are based on linear elastic fracture mechanics. Therefore, if an interface is first loaded purely in the normal deformation mode, unloaded, and subsequently loaded in a purely shear deformation mode, the mode mix ratios based on accumulated energies at the end of the above deformation path evaluate to (assuming the shear deformation to be in the local-1 direction only) $G_n \neq 0$ and $G_s \neq 0$. On the other hand, the mode mix ratios based on nonaccumulated energies evaluate to $G_n = 0$ and $G_s \neq 0$ at the end of the above deformation path.

Damage Evolution Definition

There are two components to the definition of damage evolution. The first component involves specifying either the effective separation at complete failure, δ_m^f, relative to the effective separation at the initiation of damage, δ_m^o; or the energy dissipated due to failure, G^C (see *Figure 8*). The second component to the definition of damage evolution is the specification of the nature of the evolution of the damage variable, D, between initiation of damage and final failure. This can be done by either defining linear or exponential softening laws or specifying D directly as a tabular function of the effective separation relative to the effective separation at damage initiation. The data described above will in general be functions of the mode mix, temperature, or field variables.

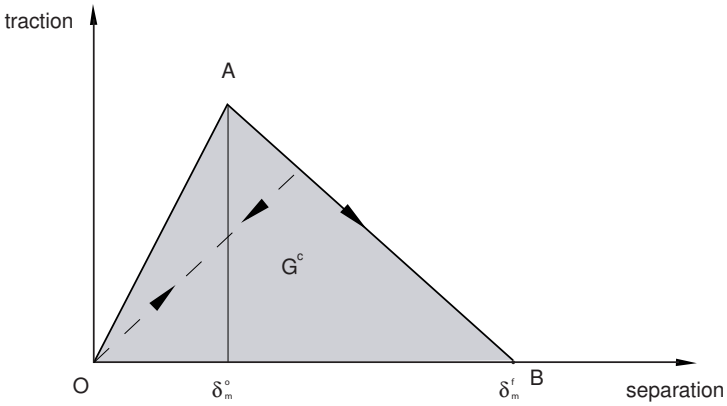


Figure 8: Linear damage evolution.

Figure 9 is a schematic representation of the dependence of damage initiation and evolution on the mode mix for a traction-separation response with isotropic shear behavior. The figure shows the traction on the vertical axis and the magnitudes of the normal and the shear separations along the two horizontal axes. The unshaded triangles in the two vertical coordinate planes represent the response under pure normal and pure shear separation, respectively. All intermediate vertical planes (that contain the vertical axis) represent the damage response under mixed-mode conditions with different mode mixes. The dependence of the damage evolution data on the mode mix can be defined either in tabular form or, in the case of an energy-based definition, analytically. The manner in which the damage evolution data are specified as a function of the mode mix is discussed later in this section.

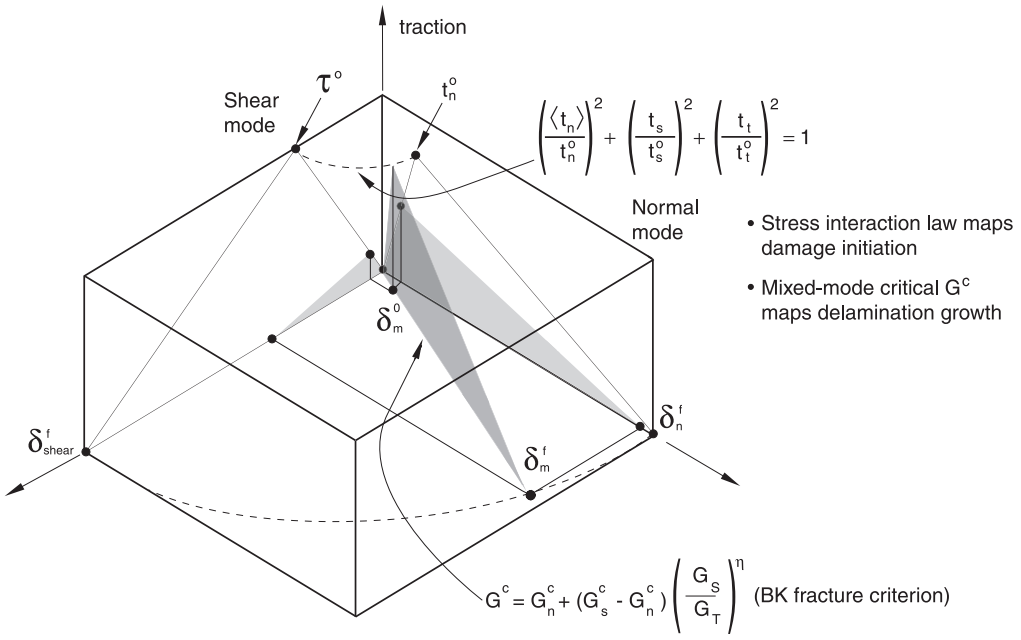


Figure 9: Illustration of mixed-mode response in cohesive interactions.

Unloading subsequent to damage initiation is always assumed to occur linearly toward the origin of the traction-separation plane, as shown in *Figure 8*. Reloading subsequent to unloading also occurs along the same linear path until the softening envelope (line AB) is reached. Once the softening envelope is reached, further reloading follows this envelope as indicated by the arrow in *Figure 8*.

Evolution Based on Effective Separation

You specify the quantity $\delta_m^f - \delta_m^o$ (i.e., the effective separation at complete failure, δ_m^f , relative to the effective separation at damage initiation, δ_m^o, as shown in *Figure 8*) as a tabular function of the mode mix, temperature, and/or field variables. In addition, you also choose either a linear or an exponential softening law that defines the detailed evolution (between initiation and complete failure) of the damage variable, D, as a function of the effective separation beyond damage initiation. Alternatively, instead of using linear or exponential softening, you can specify the damage variable, D, directly as a tabular function of the effective separation after the initiation of damage, $\delta_m - \delta_m^o$; mode mix; temperature; and/or field variables.

Linear Damage Evolution

For linear softening (see *Figure 8*) Abaqus uses an evolution of the damage variable, D, that reduces (in the case of damage evolution under a constant mode mix, temperature, and field variables) to the following expression:

$$D = \frac{\delta_m^f(\delta_m^{\max} - \delta_m^o)}{\delta_m^{\max}\left(\delta_m^f - \delta_m^o\right)}.$$

In the preceding expression and in all later references, $\delta_m^{\max}$ refers to the maximum value of the effective separation attained during the loading history. The assumption of a constant mode mix at a contact point between initiation of damage and final failure is customary for problems involving monotonic damage (or monotonic fracture).

Input File Usage: Use the following option to specify linear damage evolution:

**DAMAGE EVOLUTION*, TYPE=DISPLACEMENT,
SOFTENING=LINEAR

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damage**: **Evolution** tabbed page: **Type**: **Displacement**: **Softening**: **Linear**

Exponential Damage Evolution

For exponential softening (see *Figure 10*) Abaqus uses an evolution of the damage variable, D, that reduces (in the case of damage evolution under a constant mode mix, temperature, and field variables) to

$$D = 1 - \left\{\frac{\delta_m^o}{\delta_m^{\max}}\right\}\left\{1 - \frac{1 - \exp\left(-\alpha\left(\frac{\delta_m^{\max} - \delta_m^o}{\delta_m^f - \delta_m^o}\right)\right)}{1 - \exp\left(-\alpha\right)}\right\}.$$

In the expression above α is a nondimensional parameter that defines the rate of damage evolution and $\exp\left(x\right)$ is the exponential function.

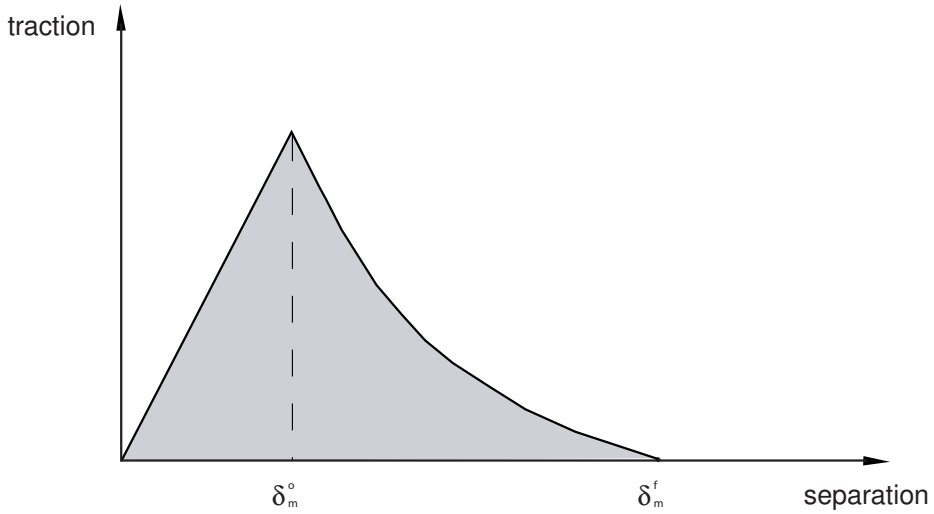


Figure 10: Exponential damage evolution.

Input File Usage: Use the following option to specify exponential softening:

**DAMAGE EVOLUTION*, TYPE=DISPLACEMENT,
SOFTENING=EXPONENTIAL

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damage**: **Evolution** tabbed page: **Type**: **Displacement**: **Softening**: **Exponential**

Tabular Damage Evolution

For tabular softening you define the evolution of D directly in tabular form. D must be specified as a function of the effective separation relative to the effective separation at initiation, mode mix, temperature, and/or field variables.

Input File Usage: Use the following option to define the damage variable directly in tabular form:

**DAMAGE EVOLUTION*, TYPE=DISPLACEMENT,
SOFTENING=TABULAR

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damage**: **Evolution** tabbed page: **Type**: **Displacement**: **Softening**: **Tabular**

Evolution Based on Energy

Damage evolution can be defined based on the energy that is dissipated as a result of the damage process, also called the fracture energy. The fracture energy is equal to the area under the traction-separation curve (see *Figure 8*). You specify the fracture energy as a property of the cohesive interaction and choose either a linear or an exponential softening behavior. Abaqus ensures that the area under the linear or the exponential damaged response is equal to the fracture energy.

The dependence of the fracture energy on the mode mix can be specified either directly in tabular form or by using analytical forms as described below. When the analytical forms are used, the mode mix ratio is assumed to be defined in terms of energies.

Tabular Form

The simplest way to define the dependence of the fracture energy is to specify it directly as a function of the mode mix in tabular form.

Input File Usage: Use the following option to specify fracture energy as a function of the mode mix in tabular form:

**DAMAGE EVOLUTION*, TYPE=ENERGY,
MIXED MODE BEHAVIOR=TABULAR

Abaqus/CAE Usage: Interaction module: contact property editor: **Contact**: **Mechanical->Damage**: **Evolution** tabbed page: **Type**: **Energy**: toggle on **Specify mixed mode behavior**: **Tabular**

Power Law Form

The dependence of the fracture energy on the mode mix can be defined based on a power law fracture criterion. The power law criterion states that failure under mixed-mode conditions is governed by a power law interaction of the energies required to cause failure in the individual (normal and two shear) modes. It is given by

$$\left\{\frac{G_n}{G_n^C}\right\}^\alpha + \left\{\frac{G_s}{G_s^C}\right\}^\alpha + \left\{\frac{G_t}{G_t^C}\right\}^\alpha = 1,$$

The mixed-mode fracture energy $G^C = G_T$ when the above condition is satisfied. In other words,

$$G^C = 1\Big/\left(\left\{\frac{m_1}{G_n^C}\right\}^\alpha + \left\{\frac{m_2}{G_s^C}\right\}^\alpha + \left\{\frac{m_3}{G_t^C}\right\}^\alpha\right)^{1/\alpha}.$$

You specify the quantities G_n^C, G_s^C, and G_t^C, which refer to the critical fracture energies required to cause failure in the normal, the first, and the second shear directions, respectively.

Input File Usage: Use the following option to define the fracture energy as a function of the mode mix using the analytical power law fracture criterion:

**DAMAGE EVOLUTION*, TYPE=ENERGY,
MIXED MODE BEHAVIOR=POWER LAW, POWER=α

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damage**: **Evolution** tabbed page: **Type**: **Energy**: toggle on **Specify mixed mode behavior**: **Power law**: α

Benzeggagh-Kenane (BK) Form

The Benzeggagh-Kenane fracture criterion (Benzeggagh and Kenane, 1996) is particularly useful when the critical fracture energies during separation purely along the first and the second shear directions are the same; i.e., $G_s^C = G_t^C$. It is given by

$$G_n^C + \left(G_s^C - G_n^C\right)\left\{\frac{G_S}{G_T}\right\}^\eta = G^C,$$

where $G_S = G_s + G_t$, $G_T = G_n + G_S$, and η is a cohesive property parameter. You specify G_n^C, G_s^C, and η.

Input File Usage: Use the following option to define the fracture energy as a function of the mode mix using the analytical BK fracture criterion:

**DAMAGE EVOLUTION*, TYPE=ENERGY,
MIXED MODE BEHAVIOR=BK, POWER=η

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damage**: **Evolution** tabbed page: **Type**: **Energy**: toggle on **Specify mixed mode behavior**: **Benzeggagh-Kenane**: η

Linear Damage Evolution

For linear softening (see *Figure 8*) Abaqus uses an evolution of the damage variable, D, that reduces to

$$D = \frac{\delta_m^f(\delta_m^{max} - \delta_m^o)}{\delta_m^{max}\left(\delta_m^f - \delta_m^o\right)},$$

where $\delta_m^f = 2G^C/T_{eff}^o$ with T_{eff}^o as the effective traction at damage initiation. δ_m^{max} refers to the maximum value of the effective separation attained during the loading history.

Input File Usage: Use the following option to specify linear damage evolution:

**DAMAGE EVOLUTION*, TYPE=ENERGY, SOFTENING=LINEAR

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damage**: **Evolution** tabbed page: **Type**: **Energy**: **Softening**: **Linear**

Exponential Damage Evolution

For exponential softening Abaqus uses an evolution of the damage variable, D, that reduces to

$$D = \int_{\delta_m^o}^{\delta_m^{max}} \frac{T_{eff}d\delta}{G^C - G_o}.$$

In the expression above T_{eff} and δ are the effective traction and separation, respectively. G_o is the elastic energy at damage initiation. In this case the traction might not drop immediately after damage initiation, which is different from what is seen in *Figure 10*.

Input File Usage: Use the following option to specify exponential softening:

**DAMAGE EVOLUTION*, TYPE=ENERGY,
SOFTENING=EXPONENTIAL

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damage**: **Evolution** tabbed page: **Type**: **Energy**: **Softening**: **Exponential**

Defining Damage Evolution Data as a Tabular Function of Mode Mix

As discussed earlier, the data defining the evolution of damage at the cohesive interface can be tabular functions of the mode mix. The manner in which this dependence must be defined in Abaqus is outlined below for mode mix definitions based on energy and traction, respectively. In the following discussion it is assumed that the

evolution is defined in terms of energy. Similar observations can also be made for evolution definitions based on effective separation.

Mode Mix Based on Energy

For an energy-based definition of mode mix, in the most general case of a three-dimensional state of separation with anisotropic shear behavior the fracture energy, G^C, must be defined as a function of $(m_2 + m_3)$ and $[m_3/(m_2 + m_3)]$. The quantity $(m_2 + m_3) = G_S/G_T$ is a measure of the fraction of the total separation that is shear, while $[m_3/(m_2 + m_3)] = G_t/G_S$ is a measure of the fraction of the total shear separation that is in the second shear direction. *Figure 11* shows a schematic of the fracture energy versus mode mix behavior.

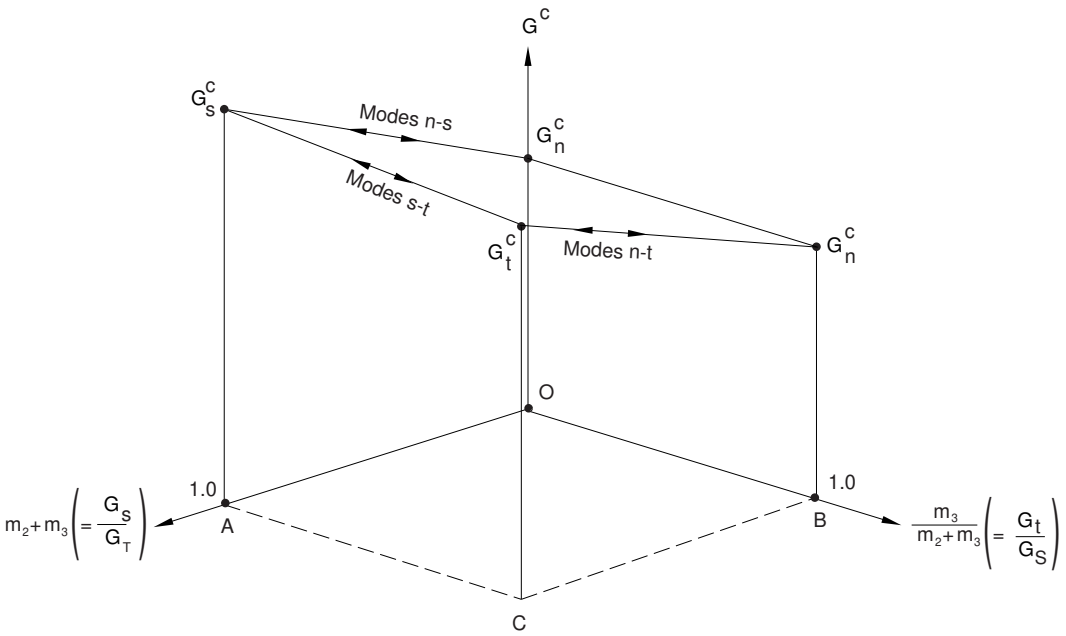


Figure 11: Fracture energy as a function of mode mix.

The limiting cases of pure normal and pure shear separations in the first and second shear directions are denoted in *Figure 11* by G_n^C, G_s^C, and G_t^C, respectively. The lines labeled “Modes n-s,” “Modes n-t,” and “Modes s-t” show the transition in behavior between the pure normal and the pure shear in the first direction, pure normal and pure shear in the second direction, and pure shears in the first and second directions, respectively. In general, G^C must be specified as a function of $(m_2 + m_3)$ at various fixed values of $[m_3/(m_2 + m_3)]$. In the discussion that follows we refer to a data set of G^C versus $(m_2 + m_3)$ corresponding to a fixed $[m_3/(m_2 + m_3)]$ as a “data block.” The following guidelines are useful in defining the fracture energy as a function of the mode mix:

- For a two-dimensional problem G^C needs to be defined as a function of m_2 ($m_3 = 0$ in this case) only. The data column corresponding to $[m_3/(m_2 + m_3)]$ must be left blank. Hence, essentially only one “data block” is needed.
- For a three-dimensional problem with isotropic shear response, the shear behavior is defined by the sum $(m_2 + m_3)$ and not by the individual values of m_2 and m_3. Therefore, in this case a single “data block” (the “data block” for $[m_3/(m_2 + m_3)] = 0$) also suffices to define the fracture energy as a function of the mode mix.

- In the most general case of three-dimensional problems with anisotropic shear behavior, several "data blocks" would be needed. As discussed earlier, each "data block" would contain G^C versus $(m_2 + m_3)$ at a fixed value of $[m_3/(m_2 + m_3)]$. In each "data block" $(m_2 + m_3)$ can vary between 0 and 1.0. The case $(m_2 + m_3) = 0$ (the first data point in any "data block"), which corresponds to a purely normal mode, can never be achieved when $[m_3/(m_2 + m_3)] \neq 0$ (i.e., the only valid point on line OB in *Figure 11* is the point O, which corresponds to a purely normal separation). However, in the tabular definition of the fracture energy as a function of mode mix, this point simply serves to set a limit that ensures a continuous change in fracture energy as a purely normal state is approached from various combinations of normal and shear separations. Hence, the fracture energy of the first data point in each "data block" must always be set equal to the fracture energy in a purely normal separation (G_n^C).

  As an example of the anisotropic shear case, consider that you want to input three "data blocks" corresponding to fixed values of $[m_3/(m_2 + m_3)] =$ 0., 0.2, and 1.0, respectively. For each of the three "data blocks," the first data point must be $(G_n^C, 0)$ for the reasons discussed above. The rest of the data points in each "data block" define the variation of the fracture energy with increasing proportions of shear separation.

Mode Mix Based on Traction

The fracture energy needs to be specified in tabular form of G^C versus ϕ_1 and ϕ_2. Thus, G^C needs to be specified as a function of ϕ_1 at various fixed values of ϕ_2. A "data block" in this case corresponds to a set of data for G^C versus ϕ_1, at a fixed value of ϕ_2. In each "data block" ϕ_1 may vary from 0 (purely normal separation) to 1 (purely shear separation). An important restriction is that each data block must specify the same value of the fracture energy for $\phi_1 = 0$. This restriction ensures that the energy required for fracture as the traction vector approaches the normal direction does not depend on the orientation of the projection of the traction vector on the shear plane (see *Figure 7*).

Dependence of Damage Evolution on the Effective Rate of Separation

The damage evolution criterion can depend on the effective rate of separation. For a discussion of filtering the effective rate of separation and interpolation with respect to the effective rate of separation, see *About Mechanical Contact Properties*.

Input File Usage: Use the following option to define rate-dependent damage evolution:

**DAMAGE EVOLUTION*, RATE DEPENDENT

Abaqus/CAE Usage: Defining rate-dependent damage evolution is not supported in Abaqus/CAE.

Viscous Regularization in Abaqus/Standard

Models exhibiting various forms of softening behavior and stiffness degradation often lead to severe convergence difficulties in Abaqus/Standard. Viscous regularization of the constitutive equations defining surface-based cohesive behavior can be used to overcome some of these convergence difficulties. This technique is also applicable to cohesive elements, fastener damage, and the concrete material model in Abaqus/Standard. Viscous regularization damping causes the tangent stiffness matrix that defines the contact stresses to be positive for sufficiently small time increments.

The approximate amount of energy associated with viscous regularization over the whole model is included in the output variable ALLCD.

Input File Usage: **DAMAGE STABILIZATION*

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Damage**: **Stabilization** tabbed page: **Viscosity coefficient**

Virtual Crack Closure Technique in Abaqus/Explicit

In Abaqus/Explicit, the surface-based cohesive behavior framework can be used to model brittle crack propagation problems based on linear elastic fracture mechanics principles. The Virtual Crack Closure Technique (VCCT) fracture criterion can be used to model crack propagation in initially partially bonded surfaces. A detailed discussion of this topic can be found in *Crack Propagation Analysis*.

The VCCT fracture criterion cannot be combined with a damage-based surface behavior of the traction-separation response. However, you can use a surface-based VCCT fracture criterion in conjunction with cohesive elements. VCCT could model brittle failure/crack propagation while the cohesive elements could model other aspects of the bonded interface such as stitches.

Input File Usage: Use the following options to enforce cohesive behavior subsequent to maximum degradation:

**COHESIVE BEHAVIOR*
**FRACTURE CRITERION*, TYPE= VCCT

Output

In addition to the standard output identifiers available in Abaqus (*Abaqus/Standard Output Variable Identifiers* and *Abaqus/Explicit Output Variable Identifiers*), the following variables have special meaning for cohesive surfaces with traction-separation behavior:

CSDMG Overall value of the scalar damage variable, D.

CSMAXSCRT This variable indicates whether the maximum contact stress damage initiation criterion has been satisfied at a contact point up to the current increment. It is evaluated as

$\mathrm{CSMAXSCRT}^i = \max\left(\mathrm{CSMAXSCRT}^{i-1}, \max\left\{\frac{\langle t_n\rangle}{t_n^o}, \frac{t_s}{t_s^o}, \frac{t_t}{t_t^o}\right\}\right)$, where i is the current increment number.

CSMAXUCRT This variable indicates whether the maximum separation damage initiation criterion has been satisfied at a contact point up to the current increment. It is evaluated as

$\mathrm{CSMAXUCRT}^i = \max\left(\mathrm{CSMAXUCRT}^{i-1}, \max\left\{\frac{\langle \delta_n\rangle}{\delta_n^o}, \frac{\delta_s}{\delta_s^o}, \frac{\delta_t}{\delta_t^o}\right\}\right)$, where i is the current increment number.

CSQUADSCRT This variable indicates whether the quadratic contact stress damage initiation criterion has been satisfied at a contact point up to the current increment. It is evaluated as

$\mathrm{CSQUADSCRT}^i = \max\left(\mathrm{CSQUADSCRT}^{i-1}, \left(\frac{\langle t_n\rangle}{t_n^o}\right)^2 + \left(\frac{t_s}{t_s^o}\right)^2 + \left(\frac{t_t}{t_t^o}\right)^2\right)$, where i is the current increment number.

CSQUADUCRT This variable indicates whether the quadratic separation damage initiation criterion has been satisfied at a contact point up to the current increment. It is evaluated as

$$\mathrm{CSQUADUCRT}^{i} = \max\left(\mathrm{CSQUADUCRT}^{i-1}, \left(\frac{\langle \delta_n \rangle}{\delta_n^o}\right)^2 + \left(\frac{\delta_s}{\delta_s^o}\right)^2 + \left(\frac{\delta_t}{\delta_t^o}\right)^2\right),$$ where i is the current increment number.

For the variables above that indicate whether a certain damage initiation criterion has been satisfied or not, a value that is less than 1.0 indicates that the criterion has not been satisfied, while a value of 1.0 indicates that the criterion has been satisfied. Each damage initiation output variable indicates the maximum value of the initiation criteria up to the current increment. For example, if a loading spike causes a peak value in a damage initiation criterion between output frames, the value of the corresponding output variable will reflect the peak value at subsequent output frames. If damage evolution is specified for this criterion, the maximum value of this variable does not exceed 1.0.

By default, general contact in Abaqus/Standard uses an internally computed tracking thickness that specifies the cutoff gap distance within which contact surfaces are tracked. Only contact constraints with gap distances smaller than the tracking thickness are considered to be active for cohesive interactions and result in nonzero values for various output quantities listed here. In cases where the default tracking thickness is insufficient, you can specify a nondefault value that extends the range over which cohesive interactions are active as well the resulting output.

References

- Benzeggagh, M. L., and M. Kenane, “Measurement of Mixed-Mode Delamination Fracture Toughness of Unidirectional Glass/Epoxy Composites with Mixed-Mode Bending Apparatus,” *Composites Science and Technology*, vol. 56, pp. 439–449, 1996.
- Camanho, P. P., and C. G. Davila, “Mixed-Mode Decohesion Finite Elements for the Simulation of Delamination in Composite Materials,” *NASA/TM-2002–211737*, pp. 1–37, 2002.

Contact Wear

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- **WEAR SURFACE PROPERTIES*
- **SURFACE PROPERTY*
- **SURFACE PROPERTY ASSIGNMENT*
- **STEP CYCLING*
- **STEP CYCLING CONTROL*
- **ADAPTIVE MESH*

Overview

Mechanical contact interactions can cause surface wear (erosion) due to various microscale mechanisms, such as in abrasion and fretting.
Such mechanisms are modeled in a continuum by phenomenological approaches that characterize the evolution of wear.

Wear modeling:

- uses variants of Archard's wear model to predict the rate of surface erosion based on wear properties assigned to surfaces and contact conditions;
- accounts for local surface wear distances in contact penetration calculations but not in underlying element calculations;
- is supported for both general contact and contact pairs in Abaqus/Standard and for general contact in Abaqus/Explicit;
- can be used together with ALE adaptive meshing in Abaqus/Explicit (see *About ALE Adaptive Meshing* and);
- can be used together with step cycling in Abaqus/Standard
- supports wear accumulation in general steps with the exceptions of material transport boundary conditions in a static procedure and general contact in a steady-state transport procedure; and
- supports only solid elements for Abaqus/Explicit.

Wear Modeling

Wear is a microscale phenomenon triggered by mechanical contact resulting in cumulative loss of surface material. One or both surfaces of a contact interface might experience wear. Wear of analytical rigid surfaces is not supported. Wear coefficients are specified as surface properties. Archard’s wear model is used such that the wear distance rate at a given location is proportional to the wear coefficient, contact stress, and slip rate. Wear often occurs over large numbers of physical cycles with wear rates gradually evolving. Efficient simulation of wear often calls for each cycle simulated to represent multiple physical cycles.

Small to moderate wear distances can be modeled accurately without ALE by treating accumulated wear distances as nodal offsets for contact penetration calculations, without affecting the underlying element calculations (as shown, for example, in *Figure 1*).

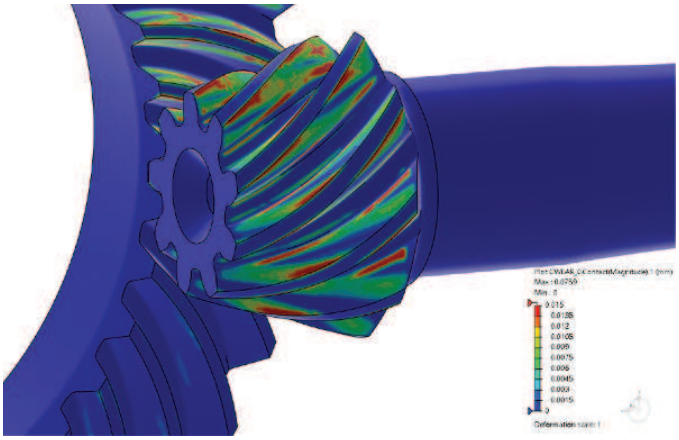


Figure 1: Simplified helical bevel gear analysis.

Wear modeling can be used together with Arbitrary Lagrangian-Eulerian (ALE) Adaptive Meshing in Abaqus/Explicit such that surface geometry reflects accumulated wear and underlying elements are smoothed, as explained in *Enabling Adaptive Meshing (ALE) for Wear in Abaqus/Explicit*. *Figure 2* demonstrates use of ALE in the context of wear modeling; in this model most wear occurs on the smaller block, with the maximum wear distance being approximately half of the original element depth. ALE mesh smoothing is effective in providing a realistic representation of the evolving part geometry and maintaining good mesh quality.

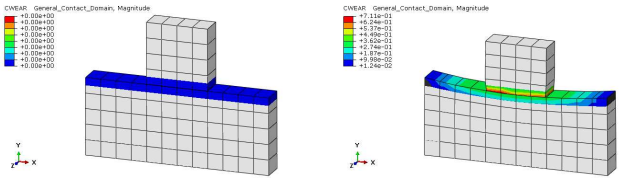


Figure 2: Effect of ALE with wear in a slider example.

Wear Rate Models

Abaqus provides four versions of Archard's wear rate model shown in *Table 1*. First, consider the expression for surface wear rate with a dimensionless wear coefficient that depends explicitly on the friction coefficient:

$$\dot{w}_s = \varphi_s \mu \sigma_p \left\| \dot{\gamma} \right\| / H_s.$$

Frictional stress is $\tau_{fric} = \mu \sigma_p$ if the (plastic) slip rate is nonzero, such that the wear rate can be written as

$$\dot{w}_s = \varphi_s \tau_{fric} \left\| \dot{\gamma} \right\| / H_s.$$

$\tau_{fric} \left\| \dot{\gamma} \right\|$ corresponds to a frictional energy density dissipation rate, $\dot{E}_{fric}$; therefore, in this case the surface wear rate is proportional to the frictional energy density dissipation rate:

$$\dot{w}_s = \frac{\varphi_s}{H_s} \dot{E}_{fric}.$$

$\dot{E}_{fric}$ is associated with contact, whereas the ratio $\frac{\varphi_s}{H_s}$ is associated with surfaces. Incremental surface wear distances might differ across an interface due to differences in $\frac{\varphi_s}{H_s}$ among surfaces; however, incremental wear for each surface of an interface is driven by local values of $\dot{E}_{fric}$ with this version of Archard's model. Of course, $\dot{w}_s$ is zero for surface locations where φ_s is zero.

Other forms of Archard's model shown in *Table 1* are provided for cases in which the damage mechanism is more closely related to contact pressure than frictional stress or for cases in which experimental data for characterizing wear coefficients is gathered without consideration of material hardness. Surface wear coefficients might depend on wear distance, contact pressure, temperature, and field variables. For example, dependence of a wear coefficient on the local wear distance can be useful for modeling wear of a part with a protective surface coating. The wear coefficient can be specified to increase after erosion of the protective coating.

In Abaqus/Standard, wear accumulation on surfaces in active contact is restricted to contact formulations where portions of surfaces come into contact. Consequently, supplementary formulations (the edge-to-surface, edge-to-edge, and vertex-to-surface formulations) in general contact do not accumulate contact wear. Linear perturbation steps also do not accumulate wear.

Table 1: Alternate versions of Archard's wear equation.

	Dimensionless wear coefficient	Lumped wear coefficient with dimensions
No friction coefficient dependence	$\dot{w}_s = \widetilde{\varphi}_{wear}\sigma_p\|\dot{\gamma}\|/H_s$	$\dot{w}_s = \overline{\varphi}_{wear}\sigma_p\|\dot{\gamma}\|$
Explicit friction coefficient dependence	$\dot{w}_s = \varphi_{wear}\mu\sigma_p\|\dot{\gamma}\|/H_s$	$\dot{w}_s = \widehat{\varphi}_{wear}\mu\sigma_p\|\dot{\gamma}\|$

Specifying Wear Properties

You can define named surface wear properties to assign them to individual surfaces. You can also specify various attributes to each wear property to choose from one of the alternative versions of Archard's wear equation.

Input File Usage: Use the following options to create a named surface property and to define the wear properties:

**SURFACE PROPERTY*, NAME=*surface_wear_property_name*
**WEAR SURFACE PROPERTIES*, UNITLESS WEAR COEF=YES or NO *(default)*, FRIC COEF DEPENDENT=YES or NO *(default)*, REFERENCE STRESS=*real_value*, DEPENDENCIES=*integer_value*

Abaqus/CAE Usage: Interaction module: **Create Interaction Property**: **Type**: **Wear**

Assigning Wear Properties

For contact pairs, assign named surface wear properties to the individual surfaces of the contact pair. Both element-based and node-based surfaces support wear property assignments.

Input File Usage: Use the following option to assign named surface wear properties to a surface:

**SURFACE*, NAME=*surface_name*, PROPERTY=*surface-wear_property_name*

Example of assigning wear properties to surfaces of a contact pair:

**SURFACE PROPERTY*, NAME=WEAR1
**WEAR SURFACE PROPERTIES*, REFERENCE STRESS=150
2E-7, , , 100
3E-7, , , 200
**SURFACE PROPERTY*, NAME=WEAR2
**WEAR SURFACE PROPERTIES*, FRIC COEF DEPENDENT=YES, UNITLESS

```
 WEAR COEF=NO
10E-6
*SURFACE, NAME=S1, TYPE=NODE, PROPERTY=WEAR1
*SURFACE, NAME=M1, TYPE=ELEMENT, PROPERTY=WEAR2
*CONTACT PAIR, INTERACTION= INTPROP
S1,M1
```

In the case of general contact, you can similarly assign named wear properties to individual surfaces. For example:

```
*SURFACE PROPERTY, NAME=WEAR1
*WEAR SURFACE PROPERTIES, REFERENCE STRESS=150
2E-7, , , 100
3E-7, , , 200
*SURFACE PROPERTY, NAME= WEAR2
*WEAR SURFACE PROPERTIES, FRIC COEF DEPENDENT=YES, UNITLESS
 WEAR COEF=NO
10E-6
*SURFACE PROPERTY ASSIGNMENT, PROPERTY=WEAR
surf_A, WEAR1
surf_B, WEAR2
surf_C, WEAR1
```

Here, surface *surf_A* is assigned surface wear property WEAR1, while surface *surf_B* is assigned surface wear property WEAR2. If surfaces *surf_A* and *surf_B*, overlap, the last wear assignment WEAR2 applies in the overlap region (similar to how overlaps are handled for other property types on the **SURFACE PROPERTY ASSIGNMENT* option).

Abaqus/CAE Usage: Interaction module: **Edit Interaction**: **General contact**: **Surface properties Tab**: choose **Global wear surface property assignment** or **Wear surface property assignment**

Accounting for Wear with Nodal Offsets in Contact Calculations

Figure 3 shows two parts in contact. The penetration distance accounting for wear, h_{pen}^{wear}, corresponds to the penetration distance of the surfaces neglecting wear, h_{pen}^{surfs}, minus the combined wear distances of the individual contact surfaces, $w = w_{surf1} + w_{surf2}$. In the absence of ALE, these wear distance offsets represent total accumulated wear. For cases in which ALE mesh smoothing occurs periodically, these wear distance offsets represent wear since the most recent ALE mesh adjustments account for wear. Modifications to contact penetrations affect contact stress calculations and, thereby, influence underlying element stresses. Abaqus uses accumulated wear distances from the previous increment to modify penetrations in the current increment.

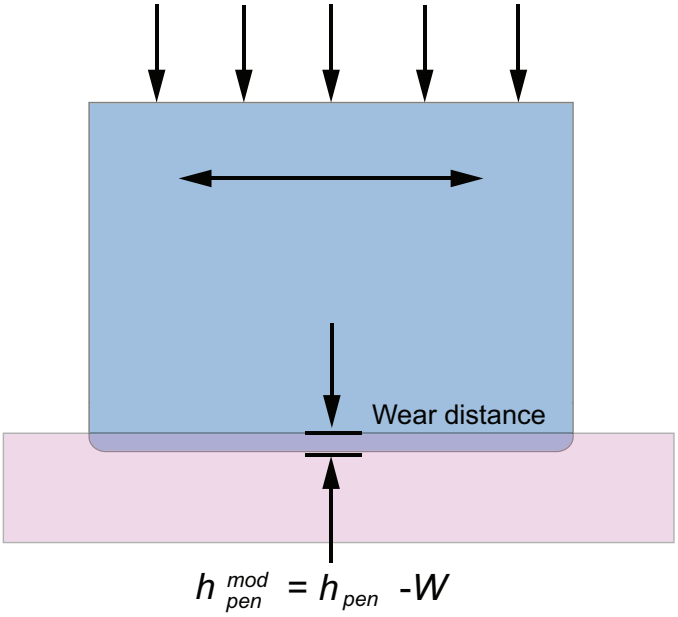


Figure 3: Contact penetration distance accounting for surface wear.

Strategies to Model Wear Accumulating Over Large Numbers of Cycles

Many wear contexts involve accumulation of wear over a large number of cycles. For example, bearings might undergo millions of rotational cycles as wear gradually accumulates during their service life. Efficient simulation of wear over a large number of physical cycles calls for each simulation cycle to approximate effects of multiple physical cycles. The ability to repeat an Abaqus step or a sequence of Abaqus steps until a stopping condition is satisfied (such as reaching a target maximum wear distance) also facilitates the simulation of wear over a large number of cycles.

Simple Strategy for Lumping Multiple Physical Wear Cycles into One Simulated Cycle

A simplistic approach to lumping “N” physical wear cycles into one simulated cycle is to include a scale factor of “N” in the specified wear coefficients. This approach is used in Abaqus/Explicit. Several sliding cycles are simulated in one Abaqus/Explicit step in this case, but each of these simulated cycles might represent N=1000 (or another number of) physical cycles. Contact stress distributions gradually evolve across wear cycles, which influences local wear rates. The number of physical wear cycles represented per simulated cycle should be small enough to integrate local wear rates across cycles accurately.

Using a Step Sequence for Modeling Wear

In Abaqus/Standard you can repeat a step or step sequence until a stopping condition is met, which is often convenient for modeling wear. *Figure 4* represents repeating a step sequence, with each step sequence representing a batch of physical wear cycles, until a stopping condition is met. Each vertical bar in *Figure 4* represents a physical wear cycle numbered along the horizontal axis. The vertical axis denotes the simulation time for each wear cycle. The heights of the vertical bars are the same because the loading history is repetitive. The solid vertical bars represent wear cycles that are simulated directly and approximate effects of a batch of physical cycles scaling the wear coefficient based on the number of physical cycles represented. The groups of dotted vertical bars next to the solid bars show batches of physical wear cycles that are not simulated directly. Depending on the batch size, only a few sample simulation cycles might be needed to make accurate wear predictions associated with a large number of physical cycles.

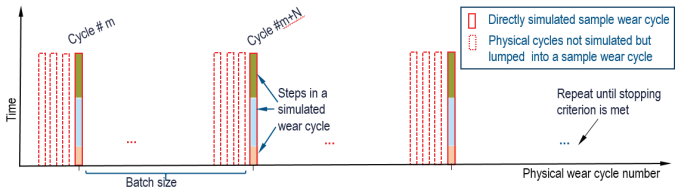


Figure 4: Physical and sample simulated wear cycles for wear accumulation.

When modeling wear with this approach, you would typically specify:

- the number of physical cycles to be represented by the first simulation cycle;
- a target value of the maximum incremental wear distance per simulated cycle; and
- a stopping condition, such as the maximum total wear.

The number of physical cycles represented by the second and subsequent simulated cycles is automatically adjusted based on the incremental wear that occurred during the previous sample simulation cycle.

When you use this approach, you should prescribe boundary conditions in rate form (such as velocities) in case the beginning and ending configurations of a simulation cycle do not match.

In the example below, a sample wear cycle is modeled with two general steps with names `WearStep-1` and `WearStep-2`. These two steps will cycle as follows:

- The step cycling sequence can be repeated up to 10 times.
- The initial batch size is 10,000 cycles.
- The upper bound on the batch size is 25,000 physical cycles.

In addition, step cycling controls are used to specify wear-based stopping conditions for the step cycling loop and target values for the maximum incremental wear distance per simulated cycle for two different node sets.

Input File Usage: Use the following to activate step cycling to repeat sample wear cycles and to control early termination and automatic calculation of batch sizes using step cycling controls:

```
*STEP CYCLING,START,NAME=Sample_cycle_repeat
10,
10000, 25000
*STEP,NAME=WearStep-1
..
..
..
*END STEP
*STEP, NAME=WearStep-2
..
..
..
*STEP CYCLING CONTROL, ACTION=END CYCLING,
CRITERION=WEAR-BASED, NAME=SCC-1
NodeSet1, AbsMin, 0.004, 0.002
NodeSet2, AbsMax, 0.012, 0.006
*END STEP
*STEP CYCLING, END
```

Wear Calculations in Steady-State Transport

The steady-state transport procedure uses a mixed Eulerian-Lagrangian method where material can convect through a stationary mesh that can deform in a Lagrangian manner with respect to a rotating frame. Consequently, the wear distance at a nodal location on the mesh is the wear a material point occupying the nodal location instantaneously accumulates as it traverses along a streamline during each revolution of the rotating frame. In other words, the wear rate history at a material point can be calculated by a spatial variation of the local wear rates along a streamline under steady-state assumptions.

The material point accumulates wear during the times when contact is active and does not accumulate any wear when not in contact during each revolution. Consequently, the wear rate is an average of the local wear rates weighted by the arc lengths on the streamline over which the local wear rates apply. This average wear rate is applied to all nodes on a streamline. For more information about the wear calculations in steady-state transport for a specific version of the Archard's wear equation, see *Wear rate calculation*.

Wear accumulation in steady-state transport can result in large wear offsets for contact penetrations and possibly affect accuracy. Also, if the increment sizes are large, the wear increments can be correspondingly large since wear accumulation in each increment at a node corresponds to the node's traversal along a streamline several times in the time increment. Since lagging wear distances are used for calculating penetrations in the current increment small increment sizes may be necessary to maintain accuracy.

Enabling Adaptive Meshing (ALE) for Wear in Abaqus/Explicit

The Arbitrary Lagrangian-Eulerian (ALE) adaptive meshing method offers a way to incorporate wear effects into underlying element calculations, especially when wear distances become significant relative to element dimensions.

For contact surfaces with wear properties, the motion of the nodes is imposed by the nodal wear distance like a spatial mesh constraint and not by the adaptive meshing algorithm. Between adaptive meshing increments (by default, every 10 increments), wear is accounted for with nodal vectors that are considered in penetration calculations; when the ALE increment is triggered, the accumulated wear distance translates into actual mesh node motion.

Output

When you request contact wear output variable CWEAR, the accumulated wear vectors at the contact surface nodes are available either as field or history output in Abaqus/Standard and as field output in Abaqus/Explicit.

Input File Usage: Use the following options to request field output:

**OUTPUT*, FIELD
**CONTACT OUTPUT*
CWEAR

Use the following options to request history output:

**OUTPUT*, HISTORY
**CONTACT OUTPUT*,NSET= *Node set name*
CWEAR

References

- Archard, J. F., "Contact and Rubbing of Flat Surfaces," *Journal of Applied Physics*, vol. 24, no. 8, pp. 981–988, 1953.

Thermal Contact Properties

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References:

- *About Contact Interactions*
- *User-Defined Interfacial Constitutive Behavior*
- *GAPCON*
- **GAP*
- **GAP CONDUCTANCE*
- **GAP HEAT GENERATION*
- **GAP RADIATION*
- **INTERFACE*
- **SURFACE INTERACTION*
- *Creating interaction properties*

Overview

Thermal interaction at the surface of a body:

- can be included in heat transfer problems (*Uncoupled Heat Transfer Analysis*, *Fully Coupled Thermal-Stress Analysis*, *Fully Coupled Thermal-Electrical-Structural Analysis*, and *Coupled Thermal-Electrical Analysis*);
- can involve conductive heat transfer between surfaces;
- can involve radiative heat transfer between surfaces when the surfaces are separated by a narrow gap;
- in Abaqus/Standard can involve convective heat flow across the boundary layer between a solid surface and a moving fluid;
- can involve heat generated by frictional work in fully coupled thermomechanical or fully coupled thermal-electrical-structural simulations; and
- in Abaqus/Standard can involve heat generated by an electrical current (Joule heating) in fully coupled thermal-electrical and fully coupled thermal-electrical-structural analyses.

General radiative heat transfer between surfaces is not discussed in this section. For information on modeling these types of problems in Abaqus/Standard, see *Cavity Radiation in Abaqus/Standard*. The thermal contact property models described here are for bodies in close proximity or in contact. For these problems gap radiation might be more efficient and robust than cavity radiation.

Thermal Properties in a Contact Property Definition

You can include all the thermal properties discussed in this section—thermal contact conductance, gap radiation, and gap heat generation—in a contact property definition for both surface-based contact and element-based contact. All three types of thermal properties can be included in the same contact property definition. Nonzero default thermal properties (which you can override) include:

- Contact conductance for touching surfaces: by default, a high value of thermal contact conductance is assigned across an interface with touching surfaces. The magnitude of this conductance is computed analogously to

the default mechanical penalty stiffness to numerically approximate a condition of matched temperature across the interface without the risk of causing overconstraints or other numerical problems.
- Dissipated energy at an interface: all dissipated energy at an interface is converted into heat by default for the gap heat generation mechanisms discussed in this section.

The thermal contact property model between two surfaces can also be completely defined through user subroutine *UINTER*, *VUINTER*, or *VUINTERACTION* (see *User-Defined Interfacial Constitutive Behavior*).

These thermal contact interaction models are intended for cases in which heat flow occurs between touching or nearby surfaces. Modeling thermal interactions over large distances with these models is often inaccurate and can significantly degrade performance.

Input File Usage: Use the following options for surface-based contact:

**SURFACE INTERACTION*, NAME=*name*
**GAP CONDUCTANCE*
**GAP RADIATION*
**GAP HEAT GENERATION*

Use the following options for element-based contact in Abaqus/Standard:

**INTERFACE* or **GAP*, ELSET=*name*
**GAP CONDUCTANCE*
**GAP RADIATION*
**GAP HEAT GENERATION*

Use the following option for user-defined, surface-based contact:

**SURFACE INTERACTION*, USER

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal**->**Thermal Conductance**, **Heat Generation**, and/or **Radiation**

Element-based contact and user-defined surface-based contact are not supported in Abaqus/CAE.

Contact Conductance between Surfaces

The conductive heat transfer between the contact surfaces is assumed to be defined by

$$q = k(\theta_A - \theta_B),$$

where q is the heat flux per unit area crossing the interface from point A on one surface to point B on the other, θ_A and θ_B are the temperatures of the points on the surfaces, and k is the thermal contact conductance. Point A is a node on the secondary surface; and point B is the location on the main surface contacting the secondary node or, if the surfaces are not in contact, the location on the main surface with a surface normal that intersects the secondary node.

Common physical behavior is such that thermal conductance across an interface is much larger while surfaces are touching (“closed” contact status) than while separated (“open” contact status). By default, Abaqus assigns a large value for k to regions in contact (independent of the contact pressure, p) and assigns $k=0$ to regions not actively in contact (independent of the contact clearance distance, d), as shown in *Figure 1*.

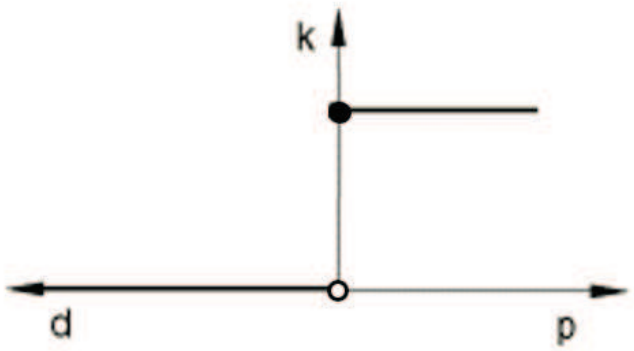


Figure 1: Default contact conductance behavior.

For heat transfer or coupled thermal-electrical analyses, the contact pressure is always zero. Therefore, contact conductance at zero contact pressure is adopted for a closed initial contact status. When the contact status is open, a contact conductance value that is a function of clearance (if provided) or a zero value is chosen.

You can define k directly or, in Abaqus/Standard, in user subroutine *GAPCON*.

Modifying Contact Conductance

The default contact conductance (shown in *Figure 1*) differs if the contact status is closed or open, but it does not depend on the contact pressure or contact clearance distance. You can modify the contact conductance for closed and open contact regimes independently and introduce dependence of the contact conductance on contact pressure and contact clearance. When defining k directly, define:

$$k = k\left(d, p, \overline{\theta}, \overline{|\dot{m}|}, \overline{f}_{\gamma}\right),$$

where

d

is the clearance between A and B,

p

is the contact pressure transmitted across the interface between A and B,

$\overline{\theta} = \frac{1}{2}\left(\theta_A + \theta_B\right)$

is the average of the surface temperatures at A and B,

$\overline{|\dot{m}|} = \frac{1}{2}\left(|\dot{m}|_A + |\dot{m}|_B\right)$

is the average of the magnitudes of the mass flow rates per unit area of the contact surfaces at A and B (this variable is not considered in an Abaqus/Explicit analysis), and

$\overline{f}_{\gamma} = \frac{1}{2}\left(f_{\gamma}^{A} + f_{\gamma}^{B}\right)$

is the average of any predefined field variables at A and B.

Modifying Contact Conductance Where the Contact Status is Closed

You can modify the contact conductance as a function of contact pressure where the contact status is closed, such as shown in *Figure 2*. When k is a function of contact pressure at the interface, the tabular data must start at zero contact pressure (or, in the case of contact that can support a tensile interface stress, the data point with the most negative pressure) and define k as p increases. The value of k remains constant for contact pressures beyond the range of data specified while contact is active. The contact conductance remains zero for separated surfaces not in contact for the examples shown in *Figure 2*. You can also modify the contact conductance for an open contact status as discussed in *Modifying Contact Conductance Where the Contact Status is Open*.

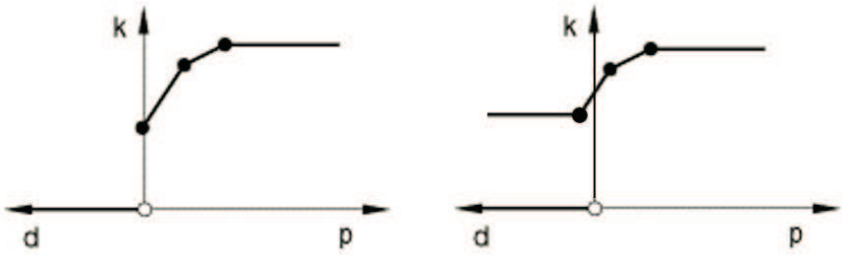


Figure 2: Examples of contact conductance, k, as a function of the contact pressure, p, with the default ($k=0$) remaining in effect where the contact status is open.

Input File Usage: **GAP CONDUCTANCE*, PRESSURE
$k, p, \overline{\theta}$

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal->Thermal Conductance**: **Definition: Tabular**, **Use only pressure-dependency data**

Modifying Contact Conductance Where the Contact Status is Open

You can modify the contact conductance as a function of the contact clearance distance, d, where the contact status is open, such as shown in *Figure 3*. Tabular data associated with k dependence on d must start at zero clearance (closed gap) and define k as the clearance distance increases. You must define at least two k versus d data points to define k as a function of the clearance. The value of k immediately drops to zero for clearance distances larger than the last data point. Therefore, there is no heat conductance when the clearance distance is greater than the value corresponding to the last data point.

If you do not also define contact conductance as a function of contact pressure, the default value of k remains in effect where the contact status is closed, as shown in *Figure 3*. *Figure 4* shows an example with the contact conductance specified as a function of contact pressure where the contact status is closed (as discussed in *Modifying Contact Conductance Where the Contact Status is Closed*) and as a function of contact clearance distance where the contact status is open.

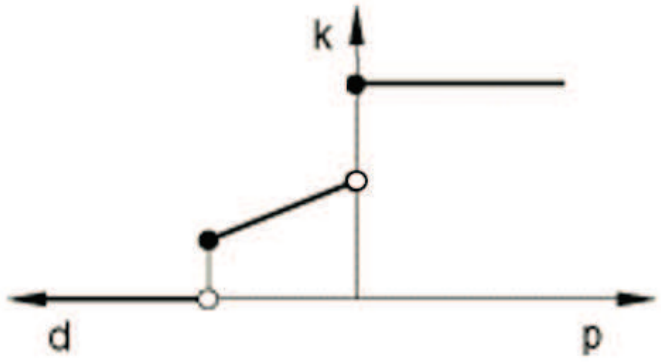


Figure 3: Example of thermal contact conductance, k, as a function of the contact clearance, d, with the default value of k remaining in effect where the contact status is closed.

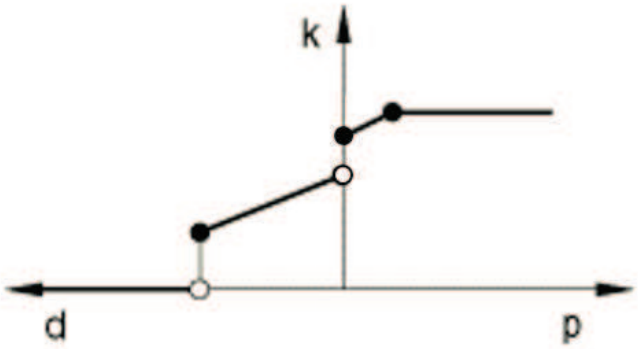


Figure 4: Example of thermal contact conductance, k, as a function of the contact pressure, p, for closed contact status and contact clearance, d, for open contact status.

Input File Usage: **GAP CONDUCTANCE*
k, d, $\overline{\theta}$

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal->Thermal Conductance**: **Definition: Tabular**, **Use only clearance-dependency data**

Using Thermal Contact Conductance to Model Convective Heat Transfer from a Surface in Abaqus/Standard

Generally, mass flow rates are defined in Abaqus/Standard (see *Forced Convection through the Mesh*) only for nodes associated with forced convection elements. However, they can be defined for any node in a model. By using the dependence of k on the average mass flow rate at the interface (in addition to other dependencies), it is possible for the contact property definition to simulate convective heat transfer to the boundary layer between a solid and a moving fluid. If mass flow rates are given only for nodes on one side of the interface, which is typically the case when simulating convective heat transfer, the average mass flow rate $\overline{|\dot{m}|}$ used to define k is half the magnitude specified.

Input File Usage: **GAP CONDUCTANCE*
k, d, $\overline{\theta}$, $\overline{|\dot{m}|}$

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal**->**Thermal Conductance**: **Definition**: **Tabular**, **Clearance Dependency** and/or **Pressure Dependency**, toggle on **Use mass flow rate-dependent data (Standard only)**

Defining Thermal Contact Conductance as a Function of Predefined Field Variables

In addition to the dependencies mentioned previously, the thermal contact conductance can depend on any number of predefined field variables, $\overline{f}_\gamma$. To make the thermal contact conductance depend on field variables, you must specify at least two data points for each field variable value.

Input File Usage: **GAP CONDUCTANCE*, DEPENDENCIES=*n*

k, *d*, $\overline{\theta}$, $\overline{|\dot{m}|}$, $\overline{f}_\gamma$

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal**->**Thermal Conductance**: **Definition: Tabular**, **Clearance Dependency** and/or **Pressure Dependency**, **Number of field variables:** *n*

Defining the Thermal Contact Conductance Using User Subroutine GAPCON

In Abaqus/Standard*k* can be defined in user subroutine *GAPCON*. In this case there is greater flexibility in specifying the dependencies of *k*. It is no longer necessary to define *k* as a function of the average of the two surface's temperatures, mass flow rates, or field variables.

$$k = k\left(d, p, \theta_A, \theta_B, |\dot{m}|_A, |\dot{m}|_B, f_\gamma^A, f_\gamma^B\right).$$

Input File Usage: **GAP CONDUCTANCE*, USER

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal**->**Thermal Conductance**: **Definition: User-defined**

Defining the Thermal Contact Conductance to Be Strongly Dependent on Temperature

If *k* depends strongly on temperature, the unsymmetric terms in the calculations start to become increasingly important in Abaqus/Standard. Using the unsymmetric matrix storage and solution scheme for the step might improve the convergence rate in the analysis (see *Defining an Analysis*).

Temperature and Field-Variable Dependence of Thermal Contact Conductance for Structural Elements

Temperature and field-variable distributions in beam and shell elements can generally include gradients through the cross-section of the element. Contact between these elements occurs at the reference surface; therefore, temperature and field-variable gradients in the element are not considered when determining thermal contact conductance, even in cases where the properties are also clearance dependent.

Thermal Contact Considerations in Abaqus/Explicit

Thermal contact conductance and gap radiation are enforced in Abaqus/Explicit with an explicit algorithm analogous to the penalty method for mechanical contact interaction. Therefore, thermal contact conductance and gap radiation can influence the stability condition; although in a fully coupled temperature-displacement analysis the mechanical portion of the system usually governs the overall stability condition (see *Fully Coupled*

Thermal-Stress Analysis). Extremely large values of thermal contact conductance or gap radiation can result in a decrease in the stable time increment, which is accounted for by the automatic time incrementation algorithm in Abaqus/Explicit.

Gap heat generation is applied within whichever algorithm (kinematic or penalty) is used to enforce the mechanical contact constraints. Gap heat generation has no effect on the stable time increment.

Thermal contact fluxes might be inaccurate during increments in which mesh adaptivity occurs if the mechanical contact constraints are enforced kinematically, because mesh adjustments occur in Abaqus/Explicit between the determination of the mechanical contact state for kinematic contact and the calculation of thermal contact fluxes. For example, mesh adjustments for adaptivity might cause discontinuity in the contact pressure: for pressure-dependent thermal contact conductance, the thermal contact conductance coefficient is set based on the pressure determined by the kinematic contact algorithm prior to the mesh adjustment, even though the thermal contact flux is applied after the mesh adjustment. The significance of this inaccuracy on the solution depends on the size and frequency of the mesh adjustments and the degree of variation in the conduction coefficient. This inaccuracy can be avoided by enforcing the mechanical contact constraints with the penalty method.

Thermal contact properties cannot be specified for general contact involving edge-to-edge contact. Thermal contact involving shell elements defined in a contact pair definition conducts heat only through the temperature degrees of freedom on the bottom of the shell (NT11) regardless of the surface definition. This can produce nonphysical heat flow if the contact is on the top of the shell. In this case it is recommended that you use general contact as the proper degrees of freedom are used depending on which side of the shell is involved in contact.

Thermal interactions can occur between surfaces within the thermal contact distance associated with the thermal contact conductance or radiation model when another surface lies between them. This can result in unrealistic behavior for multiple layers of thin shells.

Modeling Radiation between Surfaces When the Gap Is Small

Abaqus assumes that radiative heat transfer between closely spaced contact surfaces occurs in the direction of the normal between the surfaces. In models using surface-based contact this normal corresponds to the main surface normal (see *Contact Formulations in Abaqus/Standard*, *About Contact Pairs in Abaqus/Explicit*, and *About Surfaces*). In models using the contact elements available in Abaqus/Standard the element's connectivity defines the normal direction.

The gap radiation functionality in Abaqus is intended for modeling radiation between surfaces across a narrow gap. A more general capability for modeling radiation is available in Abaqus/Standard (see *Cavity Radiation in Abaqus/Standard*).

Radiative heat transfer is defined as a function of clearance between the surfaces through the effective view factor. Abaqus maintains the radiative heat flux even when the surfaces are in contact. This causes only a minor inaccuracy since normally the heat flux from conduction is much larger than the radiative heat flux.

Abaqus defines the heat flow per unit surface area between corresponding points as

$$q = C\left[\left(\theta_A - \theta^Z\right)^4 - \left(\theta_B - \theta^Z\right)^4\right],$$

where q is the heat flux per unit surface area crossing the gap at this point from surface A to surface B, θ_A and θ_B are the temperatures of the two surfaces, θ^Z is the absolute zero on the temperature scale being used, and the coefficient C is given by

$$C = \frac{F\sigma}{1/\epsilon_A + 1/\epsilon_B - 1},$$

where σ is the Stefan-Boltzmann constant, ϵ_A and ϵ_B are the surface emissivities, and F is the effective view factor, which corresponds to viewing the main surface from the secondary surface.

The view factor F must be defined as a function of the clearance, d, and should have a value between 0.0 and 1.0. The expression above accurately represents the radiation heat exchange between two infinite plates that are

close to each other, in which case the effective view factor, F equals 1.0. In all other cases, the effective view factor serves as a scaling factor used to approximate the radiation heat exchange between the two finite surfaces.

At least two pairs of $F - d$ points are required to define the view factor, and the tabular data must start at zero clearance (closed gap) and define the view factor as the clearance increases. The value of F drops to zero immediately after the last data point, so there is no radiative heat transfer when the clearance is greater than the value corresponding to the last data point (see *Figure 5*).

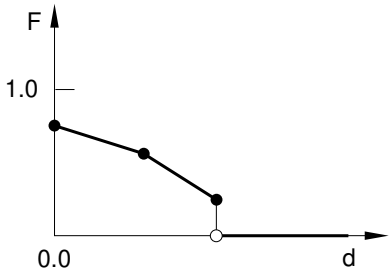


Figure 5: Example of input data to define the view factor as a function of clearance.

Input File Usage: **GAP RADIATION*
ϵ_A, ϵ_B
F_0, 0.
F_1, d_1
…

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal->Radiation**: **Emissivity of secondary surface: ϵ_A**, **Emissivity of main surface: ϵ_B**, **View factor** and **Clearance**

Specifying the Value of Absolute Zero

You must specify the value of θ^Z.

Input File Usage: **PHYSICAL CONSTANTS*, ABSOLUTE ZERO=θ^Z

Abaqus/CAE Usage: Any module: **Model->Edit Attributes->*model_name***: **Absolute zero temperature: θ^Z**

Specifying the Stefan-Boltzmann Constant

You must specify the Stefan-Boltzmann constant, σ.

Input File Usage: **PHYSICAL CONSTANTS*, STEFAN BOLTZMANN=σ

Abaqus/CAE Usage: Any module: **Model->Edit Attributes->*model_name***: **Stefan-Boltzmann constant: σ**

Improving Convergence in Abaqus/Standard

Since the heat flux due to radiation is a strongly nonlinear function of the temperature, the radiation equations are strongly nonsymmetric and using the unsymmetric matrix storage and solution scheme for the step might improve the convergence rate in Abaqus/Standard (see *Defining an Analysis*).

Modeling Heat Generated by Nonthermal Surface Interactions

In fully coupled temperature-displacement, fully coupled thermal-electrical-structural, or coupled thermal-electrical simulations, Abaqus allows for heat generation due to the dissipation of energy created by the mechanical or electrical interaction of contacting surfaces. The source of the heat in a fully coupled temperature-displacement analysis and a fully coupled thermal-electrical-structural analysis is frictional sliding; the source in a coupled thermal-electrical and a fully coupled thermal-electrical-structural analysis simulation is the flow of electrical current across the interface surfaces. By default, Abaqus releases all of the dissipated energy as heat between the surfaces and distributes it equally between the two interacting surfaces.

You can specify the fraction of dissipated energy converted into heat, η (default is 1.0), and the weighting factor, f (default is 0.5), for distribution of the heat between the interacting surfaces. η often includes a factor to convert mechanical energy into thermal energy.

$f = 1.0$ indicates that all of the generated heat flows into the first (secondary) surface of the contact pair. $f = 0.0$ indicates that all of the generated heat flows into the opposite (main) surface. Unless valid experimental data suggest otherwise, it is best to assume the default value of $f = 0.5$ because this value evenly distributes the generated heat between the surfaces. Due to surfaces often acting in both secondary and main roles for general contact, the results can be difficult to interpret when f is different than 0.5. In this case you should use the pure main-secondary formulation.

If user subroutine *UINTER*, *VUINTER*, or *VUINTERACTION* is used to define the interfacial constitutive behavior, all gap heat generation effects are turned off; you must supply an additional heat flux in the user subroutine to model these effects.

Input File Usage: **GAP HEAT GENERATION*
η, f

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal->Heat Generation**: **Specify:** η and f

Heat Generated due to Frictional Sliding

In coupled thermomechanical and coupled thermal-electrical-structural surface interactions, the rate of frictional energy dissipation is given by

$$P_{fr} = \boldsymbol{\tau} \cdot \dot{\boldsymbol{\gamma}},$$

where $\boldsymbol{\tau}$ is the frictional stress and $\dot{\boldsymbol{\gamma}}$ is the slip rate. The amount of this energy released as heat on each surface is assumed to be

$$q_A = f\eta P_{fr} \qquad \text{and} \qquad q_B = (1-f)\,\eta P_{fr},$$

where η and f are defined above. The heat flux into the secondary surface is q_A, and the heat into the main surface is q_B.

Heat Generated due to Flow of Electrical Current in Abaqus/Standard

In a coupled thermal-electrical analysis (see *Coupled Thermal-Electrical Analysis*) and a fully coupled thermal-electrical-structural analysis (see *Fully Coupled Thermal-Electrical-Structural Analysis*), the rate of electrical energy dissipated by electric current flowing across the interface is

$$P_{ec} = J\left(\varphi_A - \varphi_B\right),$$

where J is the electrical current density and φ_A and φ_B are the electrical potentials on the two surfaces. The amount of this energy released as heat on each of the interface surfaces is assumed to be

$$q_A = f\eta P_{ec} \qquad \text{and} \qquad q_B = (1-f)\,\eta P_{ec},$$

where η and f are defined in the same way as for frictional dissipation. Again, the heat flux into the secondary surface is q_A, and the heat into the main surface is q_B.

Surface-Based Interaction Variables for Thermal Contact Property Models

Abaqus provides many output variables related to the thermal interaction of surfaces. In Abaqus/Standard the values of these variables are always given at the nodes of the secondary surface. In Abaqus/Explicit these variables can be output for main and secondary surfaces, although they are not available for analytical surfaces. The variables are available only for simulations that use surface-based contact definitions. They can be requested as surface output to the data, results, or output database files (see *Surface Output from Abaqus/Standard* and *Writing Surface Output to the Output Database* for details).

Surface-Based Interaction Variables for Heat Fluxes

The following variables are available for any simulation in which heat transfer can occur (fully coupled temperature-displacement, fully coupled thermal-electrical-structural, coupled thermal-electrical, or pure heat transfer analyses):

HFL Heat flux per unit area leaving the surface.

HFLA HFL multiplied by the nodal area.

HTL Time integrated HFL.

HTLA Time integrated HFLA.

Abaqus/Standard provides all of these variables by default whenever surface output is requested to the data or results file and thermal surface interactions are present.

These variables can also be displayed in contour plots in the Visualization module of Abaqus/CAE (Abaqus/Viewer).

Surface-Based Interaction Variables for Heat Generated by Frictional Sliding

The following variables are available for fully coupled temperature-displacement simulations in which there is frictional interaction between contacting surfaces or user subroutine *UINTER*, *VUINTER*, or *VUINTERACTION* is used:

SFDR Heat flux per unit area entering the surface due to frictional dissipation (includes heat flux to both surfaces, q_A and q_B). When user subroutine *UINTER*, *VUINTER*, or

VUINTERACTION is used to define the interfacial thermal constitutive behavior, this quantity represents the heat flux resulting from the total energy dissipation due to friction and other dissipative effects. The effects of gap heat generation are turned off.

SFDRA SFDR multiplied by the nodal area.

SFDRT Time integrated SFDR.

SFDRTA Time integrated SFDRA.

WEIGHT Weighting factor, *f*, for heat flux distribution between the surfaces (available only in Abaqus/Standard; not available when the constitutive behavior of the interface is defined using user subroutine *UINTER*).

Abaqus/Standard does not provide these variables by default when surface output is requested to the data or results file; you must specify the variable identifiers.

Contour plots of these variables can also be created in the Visualization module of Abaqus/CAE (Abaqus/Viewer).

Surface-Based Interaction Variables for Heat Generated by Electrical Currents

The following variables are available for any coupled thermal-electrical and any fully coupled thermal-electrical-structural simulation:

SJD Heat flux per unit area generated by the electrical current, includes heat flux to both surfaces (q_A and q_B).

SJDA SJD multiplied by area.

SJDT Time integrated SJD.

SJDTA Time integrated SJDA.

WEIGHT Weighting factor, *f*, for heat flux distribution between the surfaces.

Abaqus/Standard does not provide these variables by default when surface output is requested to the data or results file; you must specify the variable identifiers.

Contour plots of these variables can also be plotted in the Visualization module of Abaqus/CAE (Abaqus/Viewer).

Surface-Based Interaction Variables for Heat Generated by Electrical Contact Conductance

SJDE Heat flux per unit area due to electrical current in electrolyte.

SJDEA SJDE multiplied by the nodal area.

SJDET Time integrated SJDE.

SJDETA Time integrated SJDEA.

Thermal Interaction Variables for Thermal Gap Elements

Abaqus/Standard provides the heat flux per unit area across the thermal gap elements as output. Request element output of the variable identifier HFL to the data, results, or output database file (see *Element Output* and *Writing Element Output to the Output Database* for details). The only nonzero component is HFL1: there is no heat flux tangential to the interface defined by the gap element. A positive value of HFL1 indicates heat flowing in the direction of the normal to the main surface side of the element (see *Gap Contact Elements* for the definition of this normal for DGAP elements).

Contours of the heat flux across the thermal contact elements can be plotted using Abaqus/CAE.

Thermal Interactions Involving Rigid Bodies

Various factors to consider when modeling thermal interactions involving rigid bodies are discussed in *Rigid Body Definition*. For example, Abaqus/Standard does not allow modeling of thermal interactions with analytical rigid surfaces.

Modeling Thermal Interactions with Node-Based Surfaces

The following limitations apply to fully coupled thermal-electrical-structural and fully coupled thermal-stress analyses (see *Fully Coupled Thermal-Stress Analysis*) in Abaqus/Standard:

- No heat flow occurs across a contact pair involving a node-based surface.
- No heat generation occurs for a contact pair involving a node-based surface.

These limitations do not apply to Abaqus/Explicit and do not apply to other analysis types involving thermal interactions in Abaqus/Standard (see *About Heat Transfer Analysis Procedures*).

However, when allowed, use node-based surfaces for thermal interactions with caution: Abaqus calculates the thermal interaction between bodies in terms of nodal heat fluxes that must consider the actual contact surface area associated with each node. In Abaqus/Standard this area must be specified precisely for each node in the node-based surface to calculate the correct heat fluxes; in Abaqus/Explicit a unit area is assigned to each node of a node-based surface (see *Node-Based Surface Definition*).

Thermal Interactions between Surfaces with Nodes Containing Multiple Temperature Degrees of Freedom

When the surfaces involved in a thermal interaction are defined on shell elements that have multiple temperature degrees of freedom at each node, the choice of the temperature degree of freedom at a given node for the thermal interaction depends on how the surface is defined. For an element-based surface the temperature degree of freedom closest to the surface is chosen; that is, the first temperature degree of freedom at the node for the bottom surface and the last temperature degree of freedom at the node for the top surface. For a node-based surface, the first temperature degree of freedom at the node is always chosen for a thermal interaction.

Electrical Contact Properties

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Contact Interactions*
- *Thermal Contact Properties*
- *GAPELECTR*
- **GAP ELECTRICAL CONDUCTANCE*
- **SURFACE INTERACTION*
- *Specifying gap conductance for electrical contact property options*

Overview

Electrical interactions at contact interfaces in Abaqus/Standard:

- can involve electrical conductance between solids;
- can involve electrical conductance between liquids;
- can involve ion diffusivity;
- can involve species diffusivity; and
- can involve heat generated by both forms of electrical conductance.

See *About Electrochemical Analysis Procedures*, *Coupled Thermal-Electrical Analysis*, *Fully Coupled Thermal-Electrical-Structural Analysis*, and *Modeling Solid Electrolytes and Solid-State Batteries* for details on coupled thermal-electrical and coupled thermal-electrical-structural analyses.

Electrical Properties in a Contact Property Definition

Abaqus/Standard can model the following types of electrical interactions (for degree of freedom conventions, see *Degrees of Freedom*):

- Electrical contact conductance based on the difference in the electrical potential of solids (DOF 9) across an interface.
- Electrical contact conductance based on the difference in the electrical potential of electrolyes (DOF 32) across an interface.
- Diffusion of electrical ions into and across a contact interface with consideration of ion concentrations in electrolytes (DOF 33).
- Diffusion of a species into and across a contact interface with consideration of species concentrations in the solid phase (DOF 34). For more information on the term “species,” see *Terminology*.

These electrical interaction models are intended for the transfer of electrical charge across touching or nearby surfaces. Modeling electrical interactions over large distances with these models is often inaccurate and can significantly degrade performance.

You can include electrical conductance properties in a contact property definition for surface-based contact. The electrical conductivity can be defined between solids or electrolytes.

Input File Usage: Use both of the following options to simulate electrical conductance between solids:

**SURFACE INTERACTION*, NAME=*name*
**GAP ELECTRICAL CONDUCTANCE*, TYPE=*SOLID*

Use both of the following options to simulate electrical conductance between electrolytes:

**SURFACE INTERACTION*, NAME=*name*
**GAP ELECTRICAL CONDUCTANCE*, TYPE=*LIQUID*

Abaqus/CAE Usage: Interaction module: contact property editor: **Electrical->Electrical Conductance**

Electrical Conductance between Surfaces

Abaqus/Standard models the electrical current flowing between two surfaces as

$$J = \sigma_g \left(\varphi_A - \varphi_B\right),$$

where J is the electrical current density flowing across the interface from point A on one surface to point B on the other, φ_A and φ_B are the electrical potentials on opposite points on the surfaces, and σ_g is the electrical contact conductance. Point A corresponds to a node on the secondary surface of the contact pair. Point B is the point of the main surface in contact with point A.

Common physical behavior is such that electrical conductance across an interface is much larger while surfaces are touching ("closed" contact status) than while separated ("open" contact status). By default, Abaqus assigns a large value of σ_g to regions in contact (independent of the contact pressure, p) and assigns $\sigma_g = 0$ to regions not actively in contact (independent of the contact clearance distance d), as shown in *Figure 1*.

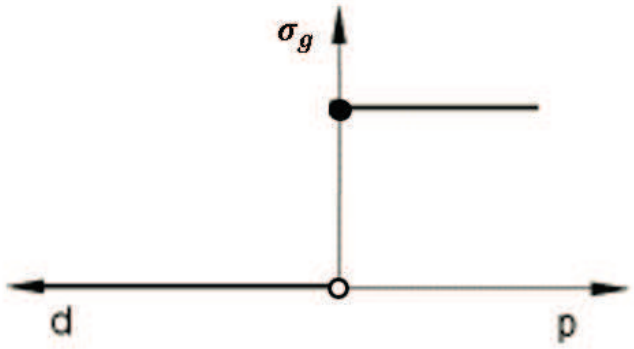


Figure 1: Default electrical conductance behavior.

For heat transfer or coupled thermal-electrical analyses, the contact pressure is always zero. Therefore, electrical conductance at zero contact pressure is adopted for a closed initial contact status. When the contact status is open, an electrical conductance value that is a function of clearance (if provided) or a zero value is chosen.

You can define the electrical conductance directly or in user subroutine *GAPELECTR*.

Modifying Electrical Conductance

The default electrical conductance (shown in *Figure 1*) differs if the contact status is closed or open, but it does not depend on the contact pressure or contact clearance distance. You can modify the electrical conductance for closed and open contact regimes independently and introduce dependence of the electrical conductance on contact

pressure and contact clearance. When you define the electrical contact conductance directly for solids, Abaqus/Standard assumes that

$$\sigma_g = \sigma_g\left(\overline{\theta}, d, p, \overline{f}^{\alpha}\right),$$

and when you define the electrical contact conductance directly for electrolytes, Abaqus/Standard assumes that

$$\sigma_g = \sigma_g\left(\overline{\theta}, \overline{C_e}, d, p, \overline{f}^{\alpha}\right).$$

Here,

$\overline{\theta} = \frac{1}{2}(\theta_A + \theta_B)$

is the average of the surface temperatures at A and B,

$\overline{C_e} = \frac{1}{2}(C_A + C_B)$

is the average of the ion concentration at A and B (only for electrolytes),

d

is the clearance between A and B,

p

is the contact pressure transmitted across the interface between A and B, and

$\overline{f}^{\alpha} = \frac{1}{2}\left(f_A^{\alpha} + f_B^{\alpha}\right)$

is the average of any predefined field variables at A and B.

Modifying Electrical Conductance Where the Contact Status is Closed

You can modify the electrical conductance as a function of contact pressure where the contact status is closed, such as shown in *Figure 2*. When σ_g is a function of contact pressure at the interface, the tabular data must start at zero contact pressure (or, in the case of contact that can support a tensile interface stress, the data point with the most negative pressure) and define σ_g as p increases. The value of σ_g remains constant for contact pressures beyond the range of data specified while contact is active. The electrical conductance remains zero for separated surfaces not in contact for the examples shown in *Figure 2*. You can also modify the electrical conductance for an open contact status as discussed in *Modifying Electrical Conductance Where the Contact Status is Open*.

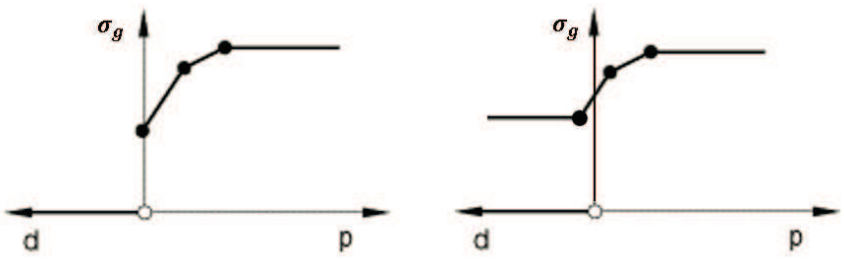


Figure 2: Examples of electrical conductance, σ_g, as a function of contact pressure, p, with default of $\sigma_g = 0$ remaining in effect where the contact status is open.

Input File Usage: **GAP ELECTRICAL CONDUCTANCE*, PRESSURE
σ_g, p, θ

Abaqus/CAE Usage: Interaction module: contact property editor: **Electrical->Electrical Conductance**; **Definition: Tabular**; **Use only pressure-dependency data**

Modifying Electrical Conductance Where the Contact Status is Open

You can modify the electrical conductance as a function of contact clearance distance, d, where the contact status is open, such as shown in *Figure 3*. Tabular data associated with σ_g dependence on d must start at zero clearance (closed gap) and define σ_g as the clearance distance increases. You must define at least two σ_g versus d data points to define σ_g as a function of the clearance. The value of σ_g immediately drops to zero for clearance distances larger than the last data point. Therefore, there is no electrical conductance when the clearance distance is greater than the value corresponding to the last data point.

If you do not also define electrical conductance as a function of contact pressure, the default value of σ_g remains in effect where the contact status is closed, as shown in *Figure 3*. *Figure 4* shows an example with electrical conductance specified as a function of contact pressure where the contact status is closed (as discussed in *Modifying Electrical Conductance Where the Contact Status is Closed*) and as a function of electrical clearance distance where the contact status is open.

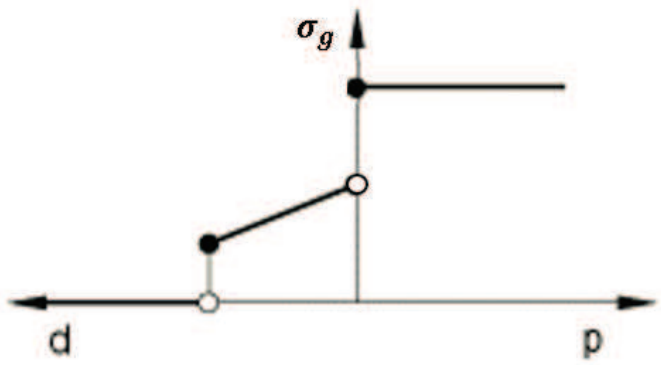


Figure 3: Example of electrical contact conductance, σ_g, as a function of the contact clearance, d, with the default value of σ_g remaining in effect where the contact status is closed.

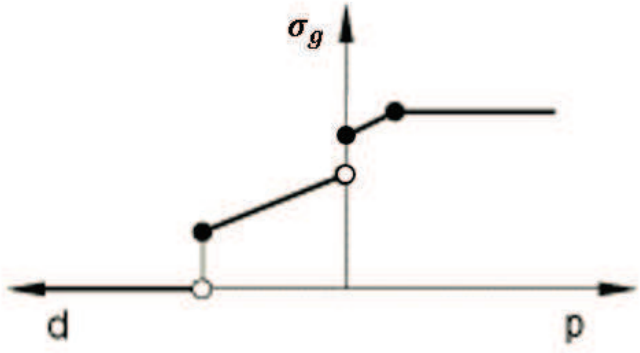


Figure 4: Example of electrical contact conductance, σ_g, as a function of the contact pressure, p, for closed contact status and as a function of contact clearance, d, for open contact status.

Input File Usage: **GAP ELECTRICAL CONDUCTANCE*
σ_g, d, θ

Abaqus/CAE Usage: Interaction module: contact property editor: **Electrical->Electrical Conductance**; **Definition: Tabular**; **Use only clearance-dependency data**

Defining Electrical Contact Conductance as a Function of Predefined Field Variables

The electrical contact conductance can be dependent on any number of predefined field variables, $\overline{f}^{\alpha}$. By default, it is assumed that the electrical conductivity depends only on the surface separation and, possibly, on the average interface temperature.

Input File Usage: **GAP ELECTRICAL CONDUCTANCE*, DEPENDENCIES=*n*

Abaqus/CAE Usage: Interaction module: contact property editor: **Electrical->Electrical Conductance**; **Definition: Tabular**, **Clearance Dependency** and/or **Pressure Dependency**, **Number of field variables:** *n*

Defining Electrical Contact Conductance Using User Subroutine GAPELECTR

When σ_g is defined in user subroutine *GAPELECTR*, there is greater flexibility in specifying the dependencies of σ_g than there is using direct tabular input. For example, it is no longer necessary to define σ_g as a function of the average of the two surfaces' temperatures or field variables:

$$\sigma_g = \sigma_g\left(\theta_A, \theta_B, d, p, f_A^{\alpha}, f_B^{\alpha}\right).$$

Input File Usage: **GAP ELECTRICAL CONDUCTANCE*, USER

Abaqus/CAE Usage: Interaction module: contact property editor: **Electrical->Electrical Conductance**; **Definition: User-defined**

Modeling Heat Generated by Electrical Conduction between Surfaces

Abaqus/Standard can include the effect of heat generated by electrical conduction between surfaces in a coupled thermal-electrical, a fully coupled thermal-electrical-structural, a coupled thermal-electrochemical analysis, a fully coupled thermal-electrochemical-structural analysis, and a fully coupled thermal-electrochemical-structural-pore pressure analysis. By default, all dissipated electrical energy is converted to heat and distributed equally between the two surfaces. You can modify the fraction of electrical energy that is released as heat and the distribution between the two surfaces; see *Modeling Heat Generated by Nonthermal Surface Interactions* for details.

Including Ion or Species Concentration Properties in a Contact Property Definition

The following discussion describes the flow of ion concentration across a contact interface; however, this discussion is equally applicable to species concentration.

Abaqus/Standard assumes that ion concentration flows in the normal direction at a contact interface and does not flow tangentially along the interface. Two contributions to the ion concentration flow into each surface at a

contact interface are generally present, as shown in *Figure 5*. The ion concentration flow into the main and secondary surfaces at corresponding points on the interface are J_M and J_S, respectively.

- One contribution (J_{across}) is associated with flow across the interface. A positive value of J_{across} corresponds to flow out from the main surface and into the secondary surface.
- The other contribution (J_{gapS} for the secondary surface and J_{gapM} for the main surface) is associated with removing or adding ion concentration from the region between the surfaces while the gap distance is changing.

  The sign convention is such that J_{gapS} and J_{gapM} are positive when these contributions flow into the respective surfaces (while the gap width decreases). The sum of J_{gapS} and J_{gapM} (which is the same as the sum of J_S and J_M) is equal to negative one times the rate of change of the gap width up to the threshold distance discussed in *Controlling the Distance within Which Ionic/Species Flux Contact Properties Are Active*.

In steady-state analyses the rate of separation of the surfaces is zero, so the ion concentration flow contributions J_{gapS} and J_{gapM} are zero; all fluid flowing out of one surface flows into the other in steady-state analyses.

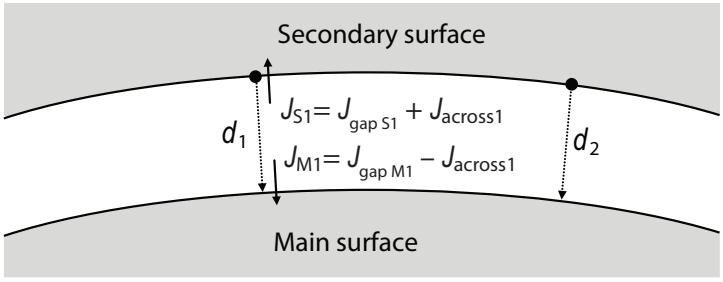


Figure 5: Flow patterns in the interface contact element.

Ion concentration flow at a contact interface typically occurs even if contact diffusivity characteristics are not specified explicitly in the contact property definition. Alternatively, you can specify contact diffusivity characteristics directly for enhanced control over the ion concentration flow across a contact interface.

Input File Usage: **SURFACE INTERACTION*, NAME=*interaction_name*
**GAP DIFFUSIVITY*

Abaqus/CAE Usage: Interaction module: contact property editor: **Electrical->Diffusivity**

Controlling the Distance within Which Ionic/Species Flux Contact Properties Are Active

The following discussion describes the method of controlling the distance over which ion concentration flux contact properties are active; however, the discussion is equally applicable to species concentration flux.

The models governing ionic flux across a contact interface are most appropriate for two surfaces in contact or separated by a relatively small gap distance. By default, Abaqus assumes that no ionic flux occurs once the surfaces have separated by a distance larger than the characteristic element length of the underlying surfaces. Alternatively, you can specify a cutoff gap distance directly beyond which no ionic flux occurs. Separate controls are provided for the contribution of ionic flux across the interface (J_{across}) and the contribution of ionic flux into the interface (J_{gap}).

Input File Usage: Use the following option to specify a cutoff distance (d_{across}) for the contribution of ionic flux across the contact interface (J_{across}):

*GAP DIFFUSIVITY, CUTOFF FLOW ACROSS=d_{across}

Use the following option to specify a cutoff distance (d_{gap}) for the contribution of ionic flux into the contact interface (J_{gap}):

*GAP DIFFUSIVITY, CUTOFF GAP FILL=d_{gap}

Abaqus/CAE Usage: Interaction module: contact property editor: **Electrical->Diffusivity**

Controlling Gap Diffusivity Associated with Ionic/Species Flux across a Contact Interface

The following discussion describes controlling gap diffusivity associated with ionic flux across a contact interface; however, the discussion is equally applicable to species flux.

If you do not specify gap diffusivity characteristics, the implied physical model is as follows:

$$C_{eA} - C_{eB} = 0\,,$$

where C_{eA} and C_{eB} are the ion concentrations at points on opposite sides of the interface. This relationship represents continuity of the ion concentration on opposite sides of a contact interface (although the condition is approximated if penalty enforcement is used—see *Controlling the Constraint Enforcement Method*) while the contact separation is less than the threshold distance discussed in *Controlling the Distance within Which Ionic/Species Flux Contact Properties Are Active*. This relationship implies that gap diffusivity across the interface is infinite.

Alternatively, you can specify a gap diffusivity, D_g, such that ionic flux across a contact interface (J_{across}, discussed above in *Including Ion or Species Concentration Properties in a Contact Property Definition*) is proportional to the difference in ion concentration magnitudes across the interface:

$$J_{across} = D_g\left(C_{eA} - C_{eB}\right).$$

When defining D_g directly, define it as

$$D_g = D_g\left(p_{contact}, \overline{C}_e, \overline{\theta}, \overline{f}_\gamma\right),$$

where

$p_{contact}$

is the contact pressure transmitted across the interface between *A* and *B*,

$\overline{C}_e = \frac{1}{2}\left(C_{eA} + C_{eB}\right)$

is the average of the ion concentrations at *A* and *B*,

$\overline{\theta} = \frac{1}{2}\left(\theta_A + \theta_B\right)$

is the average of the surface temperatures at *A* and *B*, and

$\overline{f}_\gamma = \frac{1}{2}\left(f_\gamma^A + f_\gamma^B\right)$

is the average of any predefined field variables at *A* and *B*.

Figure 6 shows an example of D_g depending on the contact pressure. Use tabular data to specify the value of D_g at one or more contact pressures as p increases. The value of D_g remains constant for contact pressures outside of the interval defined by the data points. Once the surfaces have separated, D_g remains at a constant value until the separation between the surfaces exceeds the specified flow cutoff distance (see *Controlling the Distance within Which Ionic/Species Flux Contact Properties Are Active*), at which point D_g drops to zero.

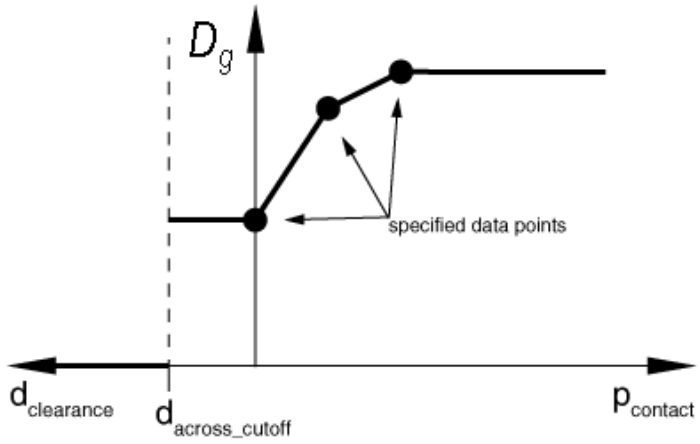


Figure 6: Contact pressure–dependent contact diffusivity.

Input File Usage: **GAP DIFFUSIVITY*

D_g, $p_{contact}$, $\overline{C}_e$, $\overline{\theta}$

Abaqus/CAE Usage: Interaction module: contact property editor: **Electrical->Diffusivity**

Defining Gap Diffusivity as a Function of Predefined Field Variables

In addition to the dependencies mentioned previously, the gap diffusivity can depend on any number of predefined field variables, $\overline{f}_\gamma$. To define gap diffusivity dependent on field variables, you must specify at least two data points for each field variable value.

Input File Usage: **GAP DIFFUSIVITY*, DEPENDENCIES=*n*

D_g, $p_{contact}$, $\overline{C}_e$, $\overline{\theta}$, $\overline{f}_\gamma$

Abaqus/CAE Usage: Interaction module: contact property editor: **Electrical->Diffusivity**

Surface-Based Output Variables for Electrical Contact Property Models

Abaqus/Standard provides the following output variables related to the electrical interaction of surfaces:

ECD	Electric current per unit area leaving secondary surface.
ECDA	ECD multiplied by the area associated with the secondary node.

ECDT	Time integrated ECD.
ECDTA	Time integrated ECDA.
ECDE	Electrical current in electrolyte per unit area.
ECDEA	ECDE multiplied by the nodal area.
ECDET	Time integrated ECDE.
ECDETA	Time integrated ECDEA.

Additional surface variables are available as described in *Ion Diffusivity of Electrolytes* and *Species Diffusivity in Solids*.

The values of these variables are always given at the nodes of the secondary surface. They can be requested as surface output to the data, results, or output database files (see *Surface Output from Abaqus/Standard* and *Writing Surface Output to the Output Database* for details).

Contour plots of these variables can also be displayed in the Visualization module of Abaqus/CAE (Abaqus/Viewer).

Pore Fluid Contact Properties

Products: Abaqus/Standard

References:

- *About Contact Interactions*
- **CONTACT PERMEABILITY*
- **SURFACE*
- **SURFACE INTERACTION*
- **CONTACT PAIR*
- **CONTACT CONTROLS ASSIGNMENT*

Overview

The pore fluid contact property models:

- are often used in geotechnical applications, where pore pressure continuity between material on opposite sides of an interface must be maintained;
- govern pore fluid flow across a contact interface and into a gap region for nearby contact surfaces;
- are applicable when pore pressure degrees of freedom are present on both sides of a contact interface (if pore pressure degrees of freedom are present on only one side of a contact interface, the surfaces are treated as impermeable);
- affect the pore fluid flow normal to the contact surfaces;
- can apply to small- and finite-sliding contact formulations; and
- assume that there is no fluid flowing tangentially to the surface.

Contact in coupled pore fluid diffusion/stress analysis involves displacement constraints to resist penetrations and pore fluid contact properties that influence the fluid flow. See *Coupled Pore Fluid Diffusion and Stress Analysis* for details on coupled pore fluid diffusion/stress analyses. See *Defining the Constitutive Response of Fluid within the Cohesive Element Gap* for details on the use of pore pressure cohesive elements as an alternative to using contact models and pore fluid contact properties.

Contact Pressure in Pore Fluid Interactions

The pore fluid contact properties discussed in this section apply when pore pressure degrees of freedom exist on both sides of a contact interface. In such cases the calculated contact pressure is effective; it does not include the pore fluid pressure contribution.

If only one side of a contact interface includes pore pressure degrees of freedom, no fluid flow into or across the contact interface occurs. In this case the reported contact pressure represents the total pressure, including the effective structural and pore fluid pressure contributions; but only the effective contact pressure is used for the computation of friction.

Including Pore Fluid Properties in a Contact Property Definition

Abaqus/Standard assumes that pore fluid flows in the normal direction at a contact interface and does not flow tangentially along the interface. Two contributions to the fluid flow into each surface at a contact interface are generally present, as shown in *Figure 1*. The fluid flow into the main and secondary surface at corresponding points on the interface are q_S and q_M, respectively.

- One contribution (q_{across}) is associated with flow across the interface. A positive value of q_{across} corresponds to flow out from the main surface and into the secondary surface.
- The other contribution (q_{gapS} for the secondary surface and q_{gapM} for the main surface) is associated with removing or adding fluid from the region between the surfaces while the gap distance is changing. The sign convention is such that q_{gapS} and q_{gapM} are positive when these contributions flow into the respective surfaces (while the gap width decreases). The sum of q_{gapS} and q_{gapM} (which is the same as the sum of q_S and q_M) is equal to negative one times the rate of change of the gap width up to the threshold distance discussed in *Controlling the Distance within Which Pore Fluid Contact Properties Are Active*.

In steady-state analyses the rate of separation of the surfaces is zero, so the fluid flow contributions q_{gapS} and q_{gapM} are zero; all fluid flowing out of one surface flows into the other in steady-state analyses.

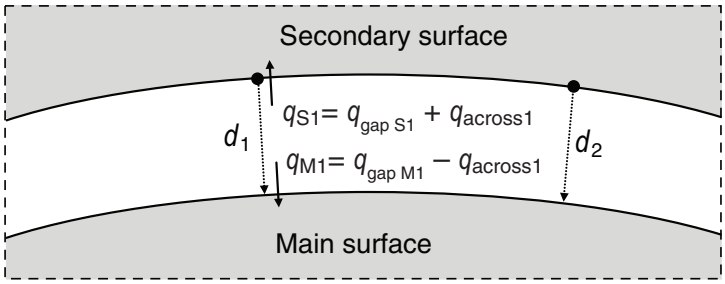


Figure 1: Flow patterns in the interface contact element.

Pore fluid flow at a contact interface typically occurs even if contact permeability characteristics are not explicitly specified in the contact property definition. Alternatively, you can directly specify contact permeability characteristics for enhanced control over the flow of fluid across a contact interface and the type of constraint enforcement method.

Input File Usage: **SURFACE INTERACTION*, NAME=*interaction_name*
**CONTACT PERMEABILITY*

Controlling the Distance within Which Pore Fluid Contact Properties Are Active

The models governing fluid flow across a contact interface are most appropriate for two surfaces in contact or separated by a relatively small gap distance. By default, Abaqus assumes no fluid flow occurs once the surfaces have separated by a distance larger than the characteristic element length of the underlying surfaces. Alternatively, you can directly specify a cutoff gap distance beyond which no fluid flow occurs. Separate controls are provided for the contribution of fluid flow across the interface (q_{across}) and the contribution of fluid flow into the interface (q_{gap}).

Input File Usage: Use the following option to specify a cutoff distance (d_{across}) for the contribution of fluid flow across the contact interface (q_{across}):

*CONTACT PERMEABILITY, CUTOFF FLOW ACROSS=d_{across}

Use the following option to specify a cutoff distance (d_{gap}) for the contribution of fluid flow into the contact interface (q_{gap}):

*CONTACT PERMEABILITY, CUTOFF GAP FILL=d_{gap}

Controlling Contact Permeability Associated with Fluid Flow across a Contact Interface

If you do not specify contact permeability characteristics, the implied physical model is continuity of the pore pressures on opposite sides of a contact interface (although the condition will be approximated if penalty enforcement is used—see *Controlling the Constraint Enforcement Method*) while the contact separation is less than the threshold distance discussed in *Controlling the Distance within Which Pore Fluid Contact Properties Are Active*:

$$p_A - p_B = 0\,,$$

where p_A and p_B are pore pressures at points on opposite sides of the interface. This relationship implies that contact permeability across the interface is infinite.

Alternatively, you can specify a contact permeability, k, such that fluid flow across a contact interface (q_{across}, discussed above in *Including Pore Fluid Properties in a Contact Property Definition*) is proportional to the difference in pore pressure magnitudes across the interface:

$$q_{across} = k\left(p_A - p_B\right).$$

When defining k directly, define it as

$$k = k\left(p_{contact}, \overline{p}_{pore}, \overline{\theta}, \overline{f}_{\gamma}\right),$$

where

$p_{contact}$

is the contact pressure transmitted across the interface between A and B,

$\overline{p}_{pore} = \frac{1}{2}\left(p_{poreA} + p_{poreB}\right)$

is the average of the pore pressures at A and B,

$\overline{\theta} = \frac{1}{2}\left(\theta_A + \theta_B\right)$

is the average of the surface temperatures at A and B, and

$\overline{f}_{\gamma} = \frac{1}{2}\left(f_{\gamma}^{A} + f_{\gamma}^{B}\right)$

is the average of any predefined field variables at A and B.

Figure 2 shows an example of k depending on the contact pressure. Use tabular data to specify the value of k at one or more contact pressures as p increases. The value of k remains constant for contact pressures outside of the interval defined by the data points. Once the surfaces have separated, k remains at a constant value until the separation between the surfaces exceeds the specified flow cutoff distance (see *Controlling the Distance within Which Pore Fluid Contact Properties Are Active*), at which point k drops to zero.

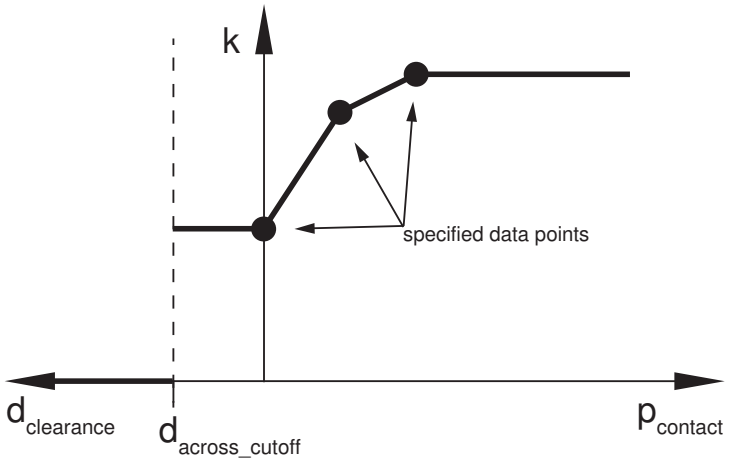


Figure 2: Contact-pressure-dependent contact permeability.

Input File Usage: **CONTACT PERMEABILITY*

$k, p_{contact}, \overline{p}_{pore}, \overline{\theta}$

Defining Gap Permeability to Be a Function of Predefined Field Variables

In addition to the dependencies mentioned previously, the gap permeability can be dependent on any number of predefined field variables, $\overline{f}_{\gamma}$. To make the gap permeability depend on field variables, at least two data points are required for each field variable value.

Input File Usage: **CONTACT PERMEABILITY*, DEPENDENCIES=*n*

$k, p_{contact}, \overline{p}_{pore}, \overline{\theta}, \overline{f}_{\gamma}$

Controlling the Constraint Enforcement Method

The default enforcement method depends on whether the contact permeability is specified. If contact permeability characteristics are not explicitly specified, the continuity of pore pressure across the interface is approximated with a penalty method (large permeability) for general contact and directly enforced for contact pairs. You can optionally specify the penalty method for contact pairs.

If contact permeability is specified, fluid flow consistent with the specific permeability is directly enforced for both contact pairs and general contact. If contact permeability is specified, the penalty method is not applicable and not allowed.

You can control the constraint enforcement through contact controls, which is the recommended method. Alternatively, you can define constraint enforcement through a surface interaction property.

Input File Usage: To directly enforce the intended physical model for contact permeability without approximation , use the following options (recommended method):

**CONTACT*
**CONTACT CONTROLS ASSIGNMENT*, TYPE=PERMEABILITY CONSTRAINT METHOD
surface_1, surface_2, DIRECT

To numerically approximate enforcement of matching pore pressures across the interface, use the following options (recommended method):

*CONTACT
*CONTACT CONTROLS ASSIGNMENT, TYPE=PERMEABILITY CONSTRAINT METHOD
surface_1, surface_2, PENALTY

To directly enforce the intended physical model for contact permeability without approximation, use the following option (alternative method):

*CONTACT PERMEABILITY, CONSTRAINT METHOD=DIRECT

To numerically approximate enforcement of matching pore pressures across the interface, use the following option (alternative method):

*CONTACT PERMEABILITY, CONSTRAINT METHOD=PENALTY

Coupled Heat Transfer–Pore Fluid Contact Properties

Heat transfer can be considered simultaneously with pore fluid flow, in which case heat flow across the contact interface can occur in conjunction with fluid flow. These various contact property aspects are defined with separate options as part of a single contact property definition that you assign to the contact interaction; see *Thermal Contact Properties* for details on defining heat transfer properties.

Output

You can write the contact surface variables associated with the interaction of contact pairs to the Abaqus/Standard data (`.dat`), results (`.fil`), and output database (`.odb`) files. In addition to the surface variables associated with the mechanical contact analysis (shear stresses, contact pressures, etc.) several pore fluid-related variables (such as pore fluid volume flux per unit area) on the contact interface can be reported. A detailed discussion of these output requests can be found in *Surface Output from Abaqus/Standard* and *Writing Surface Output to the Output Database*.

Abaqus/Standard provides the following output variables related to the pore fluid interaction of surfaces:

PFL Pore volume flux per unit area leaving the secondary surface.

PFLA PFL multiplied by the area associated with the secondary node.

PTL Time integrated PFL.

PTLA Time integrated PFLA.

TPFL Total pore volume flux leaving the secondary surface.

TPTL Time integrated TPFL.

Contact Initialization

Contact initialization controls are available for general contact and contact pairs in Abaqus/Standard and Abaqus/Explicit. In Abaqus/Standard, for small-sliding contact in both contact pairs and general contact, you can define precise initial clearance or overclosure values and contact directions for contact initialization.

In this section:

- *Contact Initialization for General Contact in Abaqus/Standard*
- *Contact Initialization for Contact Pairs in Abaqus/Standard*
- *Additional Contact Initialization Options for Small-Sliding Contact in Abaqus/Standard*
- *Contact Initialization for General Contact in Abaqus/Explicit*
- *Contact Initialization for Contact Pairs in Abaqus/Explicit*

Contact Initialization for General Contact in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About General Contact in Abaqus/Standard*
- **CONTACT INITIALIZATION ASSIGNMENT*
- **CONTACT INITIALIZATION DATA*
- *Creating contact initializations*
- *Specifying and modifying contact initialization assignments for general contact*

Overview

You can specify whether initial overclosures should be resolved without generating stresses and strains or treated as interference fits that are gradually resolved over multiple increments. You can also specify nondefault search zones that determine which nodes are affected in the case of strain-free adjustments or interference fits.

Abaqus/Standard initializes the contact state based on the gap or penetration state observed in the initial geometry. Small initial contact overclosures are resolved by default using strain-free adjustments to the positions of surface nodes. You can define alternative contact initialization methods and then assign them to contact interactions. For example, you can choose to have initial overclosures for certain interactions treated as interference fits.

Default Contact Initialization Method

By default, the general contact algorithm adjusts the initial positions of surface nodes during preprocessing to remove small initial surface overclosures without generating strains or stresses in the model, as shown in *Figure 1*. These adjustments are intended to correct only minor mismatches associated with mesh generation. Nodal adjustments are not allowed for any new contact activations at the step level. In such cases, initial overclosures are either avoided before activation or allowed to resolve by contact resulting in contact stresses.

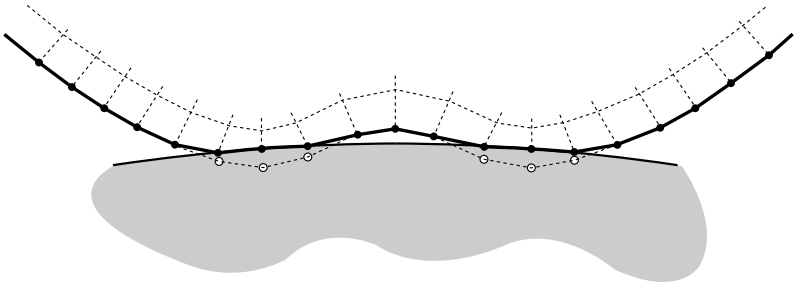


Figure 1: Configuration of contact surfaces after strain-free adjustments to resolve overclosure.

General contact automatically assigns main and secondary roles for contact interactions, as discussed in *Contact Controls Specific to General Contact in Abaqus/Standard*. Abaqus/Standard calculates an overclosure tolerance based on the size of the underlying element facets on a secondary surface. Secondary surfaces in a particular interaction are repositioned onto the associated main surface (using strain-free adjustments) if the two surfaces are initially overclosed by a distance smaller than the calculated tolerance. Initial gaps between surfaces remain unchanged by default adjustments. If a portion of a secondary surface is initially overclosed by a distance greater than the calculated tolerance, Abaqus/Standard automatically generates a contact exclusion for this surface portion and its associated main surface. Therefore, general contact does not create interactions between surfaces

(or portions of surfaces) that are severely overclosed in the initial configuration of the model, and these surfaces can freely penetrate each other throughout the analysis.

General contact uses the surface-to-surface contact formulation, so penetration/gap calculations are computed as averages over finite regions; therefore, it is possible for penetrations and gaps to be present at individual surface nodes after the adjustments. The default adjustments will not resolve initial crossings of two reference surfaces associated with shells or membranes, although techniques to resolve such cases are discussed in *Assigning Contact Initializations to Shell Surfaces*.

Defining Alternative Contact Initialization Methods

You can define alternative contact initialization methods if the default behavior is not desired. For example, you might want to increase the tolerance for deep penetrations or specify that certain openings should be adjusted to a "just touching" status. Furthermore, some analyses call for initial overclosures to be treated as interference fits rather than resolved with strain-free adjustments. To modify the contact initialization behavior, you must define one or more alternate contact initialization methods and then identify which surface pairings are to use which methods.

You assign a name to each contact initialization method. This name is used in the assignment of a contact initialization method to specific surface pairings (see *Assigning Contact Initialization Methods* below).

Input File Usage: **CONTACT INITIALIZATION DATA*, NAME=*contact_initialization_method_name*

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**: **Name:** *contact_initialization_method_name*

Increasing the Search Zones for Strain-Free Adjustments

As discussed above in *Default Contact Initialization Method*, initial gaps and large initial overclosures between surfaces are not adjusted by the default contact initialization methods. You can optionally specify nondefault search distances both above and below the surfaces in an interaction; secondary surfaces that lie within these search distances are repositioned directly onto their associated main surface using strain-free nodal adjustments. Abaqus/Standard takes shell thickness into account when calculating these search distances.

For new activations at the step level, strain-free nodal adjustments are not allowed and nodes on secondary surfaces are not repositioned. Instead, initial overclosures might have to be reconciled by precise positioning before activation while initial gaps might be resolved by establishing contact on activation with boundary motions to avoid rigid body modes. The subsequent discussion involving strain-free nodal adjustments applies only to the initial configuration.

Specifying a search distance above a surface is used to close small initial gaps between surfaces. Specifying a search distance below a surface is used to increase the default overclosure tolerance that Abaqus/Standard uses when performing strain-free adjustments; if you specify a search distance smaller than the default overclosure tolerance, Abaqus/Standard uses the default tolerance instead. As with the default initialization behavior, contact exclusions are created for initial overclosures that are larger than the specified search zone.

Increasing the extent of the search zones for strain-free adjustments can potentially increase the computational cost of an analysis. It is not generally recommended that you specify a large search zone since this might cause mesh distortion when nodes are repositioned over large distances.

Input File Usage: **CONTACT INITIALIZATION DATA*, SEARCH ABOVE=*a*, SEARCH BELOW=*b*

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**; **Abaqus/Standard contact initialization**; **Resolve with strain-free adjustments**: **Ignore overclosures greater than:** *b*, **Ignore initial openings greater than:** *a*

Specifying an Initial Clearance Distance

By default, the strain-free adjustments discussed above adjust initial nodal positions such that surfaces are "just-touching" (with zero penetration/separation). Alternatively, Abaqus/Standard can make the adjustments to achieve an initial clearance distance that you specify. The adjustments occur only for regions that satisfy the search zone tolerances, as discussed above. Mesh distortion can occur if large strain-free adjustments are required to achieve the specified initial clearance distance. Because nodal adjustments are not allowed for new activations in a step, this specification is ignored at the step level.

Input File Usage: **CONTACT INITIALIZATION DATA*, INITIAL CLEARANCE=*h*

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**; **Abaqus/Standard contact initialization**; **Specify clearance distance:** *h*

Alternative General Contact Initialization Methods Based on Storing Offsets

You can specify an initial clearance to apply to general contact interactions in portions of the model where the small-sliding tracking approach is active or to interactions in heat transfer and coupled electrical-thermal procedures that have no mechanical degrees of freedom. Abaqus/Standard stores the difference between the specified clearance and the actual clearance (with or without nodal adjustments) as offsets. In addition, for small sliding you can also specify applicable contact directions during initialization.

Storing Offsets Instead of the Default Strain-Free Adjustments

By default, Abaqus/Standard resolves clearances and overclosures in general contact by adjusting the nodal coordinates without creating strain in the model. Alternatively, you can specify unresolved gaps and overclosures at the beginning of an analysis to store as offsets. In this case, all gaps within the search distance above the secondary surface and overclosures within the search distance below the secondary surface start out as a specified zero clearance. Abaqus/Standard ignores the nondefault specification for contact interactions with the finite-sliding tracking approach, except for permanent cohesive bonds.

In situations where you specify an interference distance or an initial clearance distance, nodal adjustments are necessary to satisfy these specified initial gap or overclosure distances. In these situations, the specification to store additional offsets is ignored.

Input File Usage: **CONTACT INITIALIZATION DATA*, ADJUST=NO, SEARCH ABOVE=*a*, SEARCH BELOW=*b*

Abaqus/CAE Usage: Storing offsets for contact initialization is not supported in Abaqus/CAE.

Specifying Initial Clearance and/or Contact Directions Using Clearance Definitions for Small-Sliding Contact

You can refer to a named clearance definition to precisely specify initial clearances and/or contact directions. The different modes of specifying initial clearance using a named clearance definition carry over to defining contact initialization for general contact interactions in portions of the model where the small-sliding tracking approach is active or for interactions in heat transfer and coupled electrical-thermal procedures that have no

mechanical degrees of freedom. However, such a specification is ignored for finite-sliding contact. When node numbers or node sets are specified in a named clearance definition and they are not associated with surfaces to which the contact initialization is eventually assigned, an error message is issued. A typical use case involves referring to a named clearance and specifying that nodal adjustments should be avoided.

Input File Usage: **CONTACT INITIALIZATION DATA*, INITIAL CLEARANCE=*clearanceName*, ADJUST=NO
**CLEARANCE*, NAME=*clearanceName*, TABULAR

Abaqus/CAE Usage: Specifying initial clearance and/or contact directions for contact initialization is not supported in Abaqus/CAE.

Modeling Interference Fits

Optionally, the general contact algorithm in Abaqus/Standard can treat overclosures as interference fits. The general contact algorithm uses a shrink-fit method to gradually resolve the interference distance over the step of the analysis (if multiple load increments are used for the step) as shown in *Figure 2*, such that the fraction of the interference resolved up to and including a particular increment approximately corresponds to the fraction of the step completed. Stresses and strains are generated as the interference is resolved. Gradually resolving interference over several increments improves robustness (compared to always resolving the full interference in the first increment, which is the default for contact pairs) for cases in which a nonlinear response occurs for "interference-fit loading." It is generally recommended that you do not apply other loads while the interference fit is being resolved.

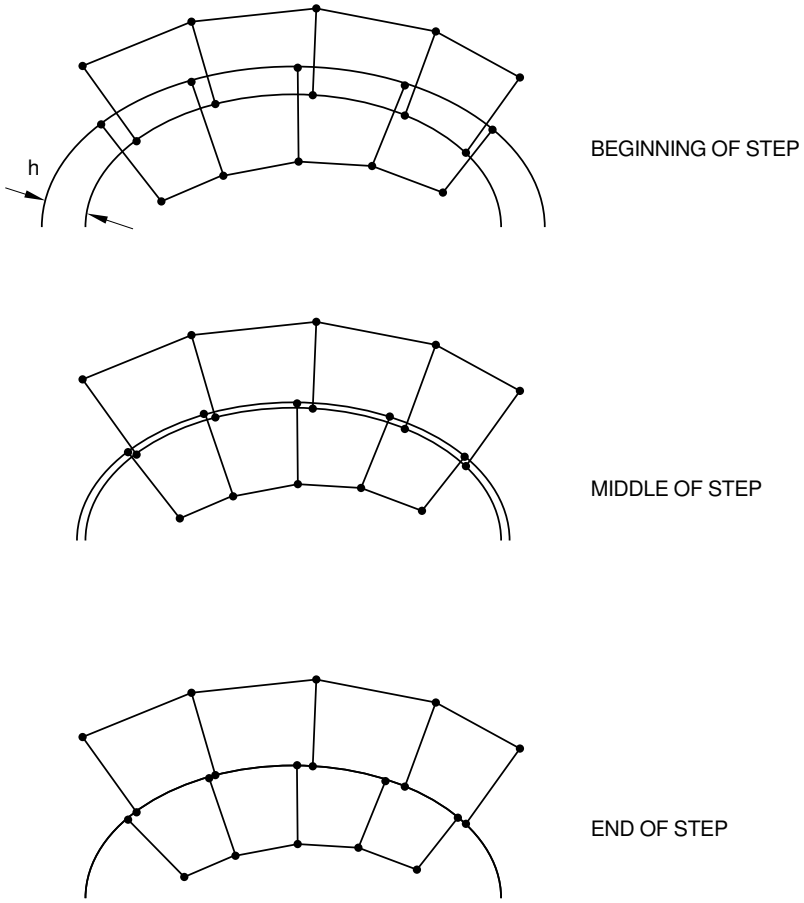


Figure 2: Gradual resolution of contact interference fit.

Because contact conditions are enforced in an average sense in a region around each constraint location for the surface-to-surface contact formulation used by general contact in Abaqus/Standard, penetrations or gaps might be observed at secondary nodes when surface-to-surface constraints are in a zero-penetration state.

Input File Usage: **CONTACT INITIALIZATION DATA*, INTERFERENCE FIT

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**; **Abaqus/Standard contact initialization**; **Treat as interference fits**

Increasing the Tolerance for Interference Fits

Abaqus/Standard calculates an overclosure tolerance based on the size of the underlying element facets on a secondary surface (see *Default Contact Initialization Method* above). An interference fit between two surfaces affects only those secondary surfaces that are overclosed by a distance smaller than the calculated tolerance; contact is ignored entirely for surfaces that are overclosed by a distance greater than the calculated tolerance.

Optionally, you can redefine the overclosure tolerance to include larger overclosures in the interference fit. If you specify a tolerance that is smaller than the default calculated tolerance, Abaqus/Standard uses the default calculated tolerance instead.

Input File Usage: **CONTACT INITIALIZATION DATA*, INTERFERENCE FIT,
SEARCH BELOW=*b*

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**; **Abaqus/Standard contact initialization**; **Treat as interference fits**: **Ignore overclosures greater than:** *b*

Specifying the Interference Distance

By default, the interference distance is implied by the initial overclosure of the mesh in the initial configuration or when surfaces are newly activated in a step; alternatively, you can specify the interference distance in the initial configuration. In this case Abaqus/Standard first makes strain-free adjustments of nodal positions such that the initial overclosure in the adjusted configuration corresponds to the specified interference distance and then invokes the shrink fit method discussed above, as depicted in *Figure 3*. Mesh distortion can occur if large strain-free adjustments are required to achieve the specified interference distance. Because strain-free nodal adjustments are not allowed for new contact activations at the step level, the interference distance specification is ignored; instead, the physical overclosure is resolved.

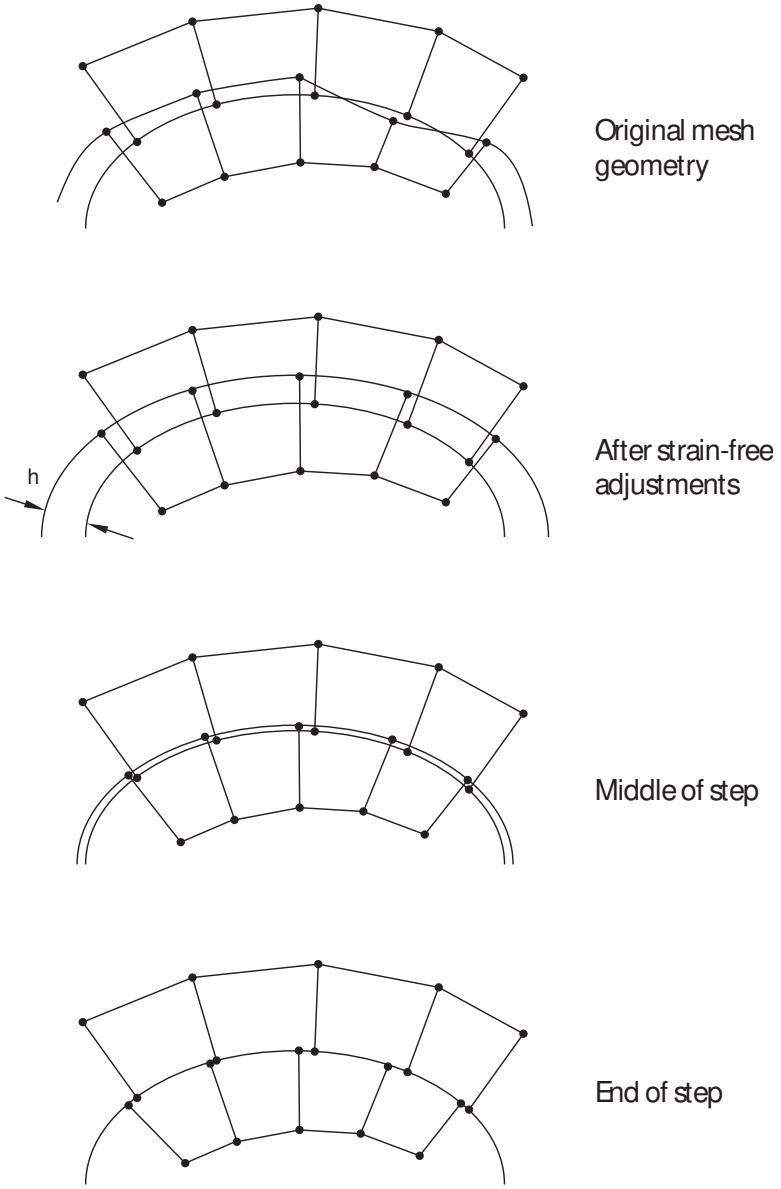


Figure 3: Treatment of a specified interference distance that differs from the interference implied by the original mesh.

The search region for the strain-free adjustments and subsequent shrink fit resolution is at least as large as the search region for the case discussed previously in which the interference distance is not specified. The search region will include overclosures at least as large as the specified interference fit and openings at least as large as the optionally specified search distance above a surface.

Input File Usage: **CONTACT INITIALIZATION DATA*, INTERFERENCE FIT=*h*, SEARCH ABOVE=*a*, SEARCH BELOW=*b*

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**; **Abaqus/Standard contact initialization**; **Treat as interference fits**: **Specify interference distance:** *h*: **Ignore overclosures greater than:** *b*, **Ignore initial openings greater than:** *a*

Deactivating Friction While Resolving Interference Fits

The presence of a friction model can degrade the robustness of resolving interference fits. It is generally recommended that you temporarily deactivate friction models while Abaqus/Standard resolves interference fits. You can deactivate the friction model in the step while interference fits are resolved using the "change friction" method discussed in *Changing Friction Properties during an Abaqus/Standard Analysis*.

Cases in Which Interference Fit Resolution with Contact Pairs Is Preferred

Large interferences might be difficult to resolve with the surface-to-surface formulation. Using this formulation, overclosures tend to be resolved along the secondary facet normal directions; using the node-to-surface formulation, which is available only with the contact pair algorithm, overclosures tend to be resolved along the main surface normal directions. *Figure 4* illustrates a case where differing normal directions lead to undesirable tangential motion during an interference fit. In some cases it might be preferable to resolve large initial overclosures with node-to-surface discretization using the contact pair algorithm (see *Modeling Contact Interference Fits in Abaqus/Standard*).

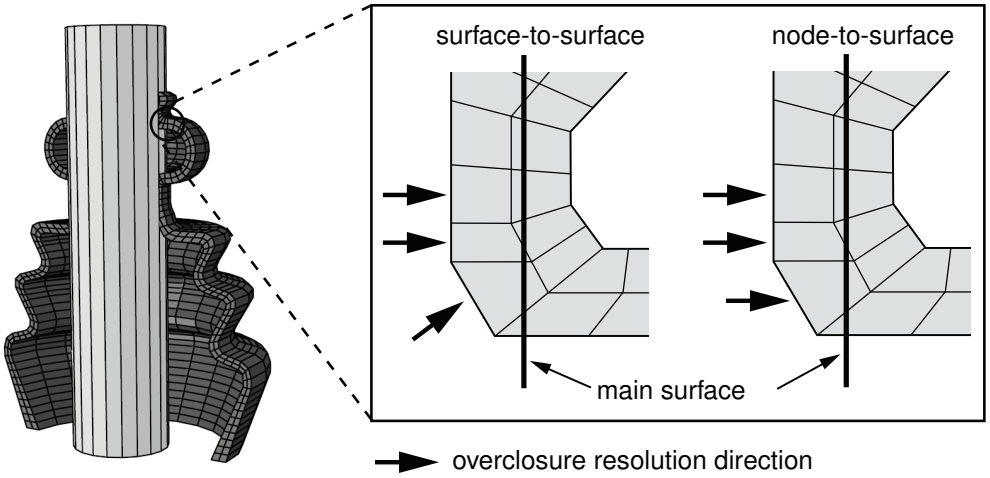


Figure 4: Comparison of contact formulations in an example with a large interference fit.

Assigning Contact Initialization Methods

You can assign contact initialization methods to selected surface pairings.

The surface names used in the assignment of contact initialization methods do not have to correspond to the surface names used to specify the general contact domain. In many cases nondefault contact initialization methods are assigned to a subset of the overall general contact domain. Any contact initialization assignments for regions that fall outside of the general contact domain are ignored. The last assignment takes precedence if the specified interactions overlap. In the case of step-dependent contact activation, contact initialization specified in a new step takes precedence over the methods specified in previous steps or at the model level.

Input File Usage: Use the following option to assign contact initialization methods:

**CONTACT INITIALIZATION ASSIGNMENT*
surface_1, *surface_2*, *contact_initialization_method_name*

This option must be used in conjunction with the **CONTACT* option. The data line can be repeated as often as necessary to assign contact initialization methods to different regions.

If the first surface name is omitted, a default surface that encompasses the entire general contact domain is assumed. If the second surface name is omitted or is the same as the first surface name, contact between the first surface and itself is assumed. Keep in mind that surfaces can be defined to span multiple unattached bodies, so self-contact is not limited to contact of a single body with itself.

If the contact initialization method name is omitted, the default contact initialization method in Abaqus/Standard is assumed. If a contact initialization method name is specified, it must also appear as the value of the NAME parameter on a **CONTACT INITIALIZATION ASSIGNMENT* option.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Contact Properties**: **Initialization assignments: Edit**: select the surfaces and the initialization in the columns on the left, and click the arrows in the middle to transfer them to the list of contact initialization assignments

Assigning Contact Initializations to Shell Surfaces

The surfaces in a contact initialization assignment can be either single- or double-sided. Single-sided surfaces must have consistent surface normal orientations for adjacent faces. Strain-free adjustments will not move surface nodes past the reference surface of the opposing surface if the assignment of a contact initialization method is made with double-sided surfaces.

Using single-sided surfaces in the assignment of a contact initialization method for shells or membranes provides enhanced control over contact initialization for cases in which shell or membrane reference surfaces are initially crossed or are initially on the wrong side of each other. *Figure 5* shows examples of adjustments for nearby segments of shell surfaces. For the case shown on the left it is assumed that single-sided surfaces with normal directions pointing away from each other are used in the assignment of the contact initialization method. In this case nodes are moved across the opposing reference surface during the strain-free adjustments.

For the case shown on the right in *Figure 5* it is assumed that single-sided surfaces with normal directions pointing toward each other are used in the assignment of the contact initialization method. In this case an initial gap is observed between the single-sided surfaces (which is also the case if double-sided surfaces are used in the contact initialization assignment). No strain-free adjustments will be made by default for openings such as this; however, if a nondefault contact initialization method is specified with an initial opening search tolerance set to a value exceeding the initial separation distance, strain-free adjustments will close the gap as shown in the figure (without moving nodes past the opposing reference surface).

Examples

The following contact initialization assignments are specified below as model or step data in a general contact analysis with the corresponding contact initialization data specified once at the model level:

- a global assignment of `shrink_fit` to the entire general contact domain;
- a local assignment of `shrink_fit_local` to contact between surfaces `surface_A` and `surface_B`—the search zone is specified explicitly to increase the default overclosure tolerance;
- a local assignment of the default Abaqus contact initialization method to contact between `surface_C` and `surface_D`; and
- a local assignment of `sfa_pickside` to contact between double-sided surfaces `surface_1` and `surface_2` by specifying one side of each surface, `surface_1_TOP` and `surface_2_BOTTOM`, in the data lines (see lower left of *Figure 5*).

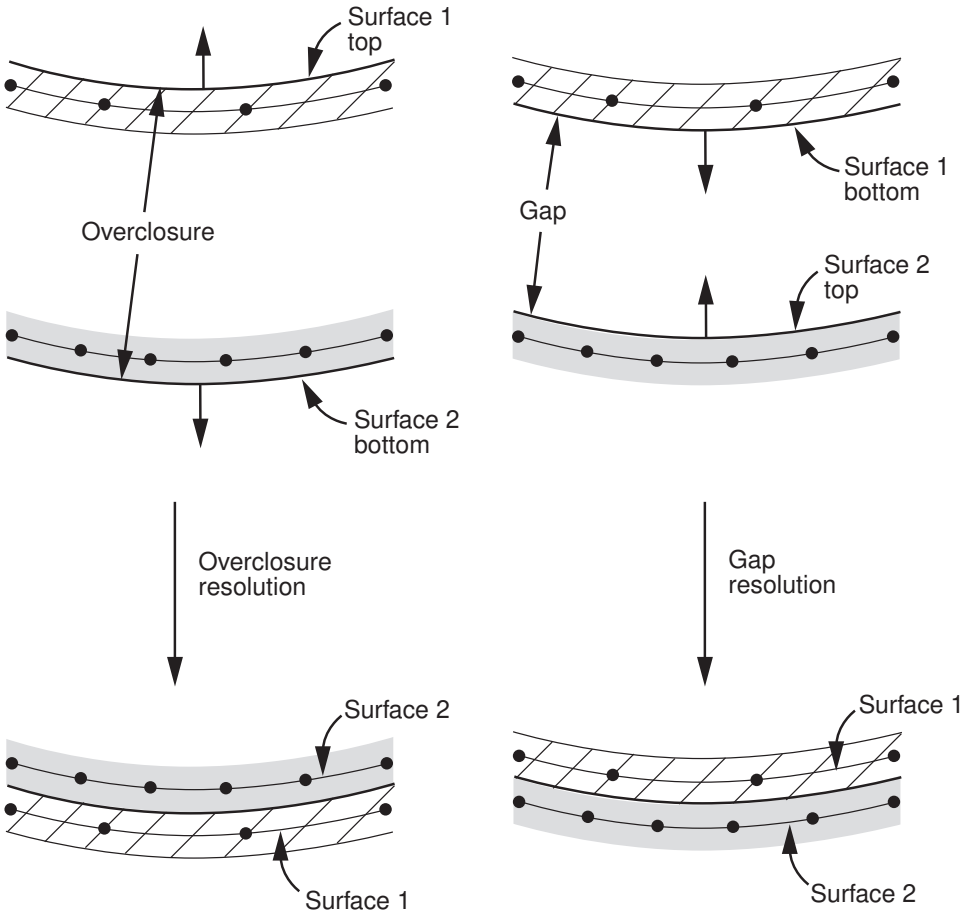


Figure 5: Strain-free adjustments during contact initialization for single-sided shell surfaces.

```
*CONTACT INITIALIZATION DATA, NAME=shrink_fit, INTERFERENCE FIT
*CONTACT INITIALIZATION DATA, NAME=shrink_fit_local,
   INTERFERENCE FIT, SEARCH BELOW = 15.0
*CONTACT INITIALIZATION DATA, NAME=sfa_pickside,
   SEARCH BELOW = 10.0
…
*CONTACT
*CONTACT INCLUSIONS, ALL EXTERIOR
*CONTACT INITIALIZATION ASSIGNMENT
 , , shrink_fit
surface_A, surface_B, shrink_fit_local
surface_C, surface_D,
surface_1_TOP, surface_2_BOTTOM, sfa_pickside
```

Contact Initialization for Contact Pairs in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Contact Pairs in Abaqus/Standard*
- *Modeling Contact Interference Fits in Abaqus/Standard*
- *Defining Tied Contact in Abaqus/Standard*
- *Contact Formulations in Abaqus/Standard*
- *Alternative General Contact Initialization Methods Based on Storing Offsets*
- **CLEARANCE*
- **CONTACT PAIR*
- *Defining surface-to-surface contact*
- *Using contact and constraint detection*

Overview

Initial overclosures in an Abaqus/Standard contact pair are treated as interference fits by default. Alternatively, initial overclosures can be treated as unintended and resolved by either adjusting the position of the surfaces automatically or, for small-sliding and tied contact pairs, introducing an offset distance term in the contact penetration calculations.

Options for modeling interference fits are discussed in *Modeling Contact Interference Fits in Abaqus/Standard*. You can also invoke an algorithm to adjust initial surface positions to close small gaps or to achieve a specified initial clearance.

Adjusting the position of surfaces in an Abaqus/Standard contact pair:

- can be performed only at the start of a simulation;
- causes Abaqus/Standard to move the nodes of the secondary surface so that they precisely contact the main surface (with some exceptions for surface-to-surface discretization and overlapping interaction definitions);
- does not create any strain in the model;
- can eliminate small gaps or penetrations caused by numerical roundoff when a graphical preprocessor such as Abaqus/CAE is used and, thus, prevent possible convergence problems;
- is required when two surfaces are tied together for the duration of the analysis;
- should not be used to correct gross errors in the mesh design;
- cannot be used with symmetric main-secondary contact; and
- will account for shell and membrane thicknesses and shell offsets (these factors are accounted for in the adjustment zone and in the adjustments) for contact formulations other than the default finite-sliding, node-to-surface contact formulation (see *Contact Formulations in Abaqus/Standard*).

Treatment of Initial Overclosures and Unintended Initial Gaps

By default, initial overclosures for Abaqus/Standard contact pairs are treated as interference fits (see *Modeling Contact Interference Fits in Abaqus/Standard*). Optionally, initial overclosures can be treated as initially in

contact and just touching the main surface. Situations in which regions with small initial overclosures or gaps should be treated as initially in contact and just touching include:

- Regions meant to fit together (be in conforming contact) in the original configuration.
- Regions meant to be tied together for the duration of the analysis (see *Defining Tied Contact in Abaqus/Standard*).

Small penetrations or gaps for nodes to treat as initially in contact can be resolved by either adjusting the position of the surfaces automatically (see *Adjusting Initial Surface Positions to Resolve Small Initial Gaps or Overclosures*) or, for small-sliding and tied contact pairs, introducing an offset distance term in the contact penetration calculations (see *Storing Penetration Offsets for Noncompliant Surfaces to Be Initially in Contact*).

Adjusting Initial Surface Positions to Resolve Small Initial Gaps or Overclosures

You can control which secondary nodes can have their initial position adjusted, as necessary, such that the initial penetration becomes zero by specifying an "adjustment zone" distance or a node set label.

Using an "Adjustment Zone" to Control Which Secondary Nodes Have Initial Gaps and Overclosures Resolved

When you specify a, the depth of the "adjustment zone," Abaqus/Standard forms an adjustment zone extending a distance a from the main surface in the gap direction and deeply into the penetration region. Abaqus/Standard measures the distance along the main surface normals that pass through the nodes of the secondary surface. Any nodes on the secondary surface that are within the "adjustment zone" in the initial geometry of the model are moved precisely onto the main surface. The motion of these secondary nodes does not create any strain in the model; it is treated as a change in the model definition. An example of adjusting the surfaces of a contact pair is shown in *Figure 1* and *Figure 2*. If you specify a negative value for a, Abaqus/Standard issues an error message.

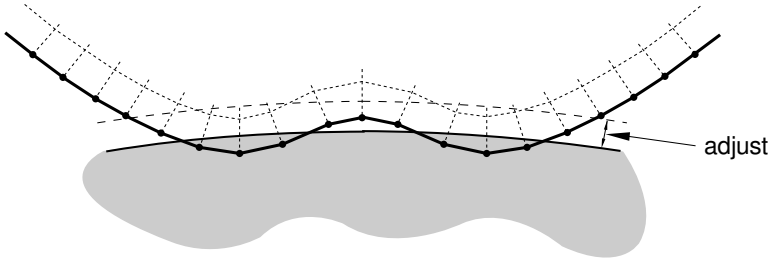


Figure 1: Initial configuration of the contact surfaces showing the "adjustment zone." The secondary surface is in bold.

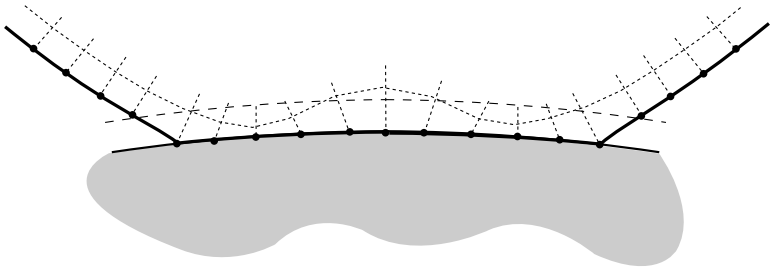


Figure 2: Configuration of the contact surfaces after the adjustment. Nodes within the adjustment zone and overclosed nodes have been moved.

Input File Usage: Use one of the following options to specify the adjustment zone:

**CONTACT PAIR*, ADJUST=*a*
secondary_surface, main_surface
...

**CONTACT PAIR*, ADJUST=YES, POSITION TOLERANCE=*a*
secondary_surface, main_surface
...

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Specify tolerance for adjustment zone**: *a*

Special Case of Adjusting Only Overclosed Secondary Nodes

When you specify the depth of the adjustment zone, Abaqus/Standard moves any secondary nodes penetrating the main surface in the initial configuration so that they just contact the main surface. Specifying a value of 0.0 for *a* causes Abaqus/Standard to adjust only those secondary nodes that are penetrating the main surface. *Figure 3* shows the effect of specifying *a*=0.0 in the example shown in *Figure 1*. Specifying *a*=0.0 is not the same as omitting the adjustment zone parameter. If you omit the adjustment zone parameter, overclosures are treated as interference fits.

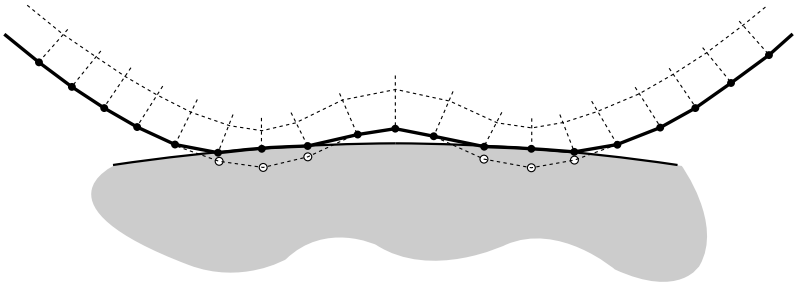


Figure 3: Adjusted configuration of contact surfaces when *a*=0.

Using a Node Set Label When Adjusting Surfaces

You can specify a node set label instead of an adjustment zone depth when only a subset of the secondary nodes should be adjusted and specifying *a* may cause the inappropriate adjustment of other secondary nodes. Abaqus/Standard adjusts only those nodes on the secondary surface belonging to the node set. The node set can contain nodes that are not on the secondary surface at all: Abaqus/Standard will ignore them and adjust only the nodes in the node set that are part of the secondary surface.

Abaqus/Standard moves any secondary nodes in the specified node set regardless of how far they are from the main surface. The adjustments of the nodes from their initial configurations do not create strains in the elements forming the secondary surface. If Abaqus/Standard adjusts secondary nodes that are far from the main surface, the elements may become poorly shaped, which can cause convergence difficulties.

Input File Usage: **CONTACT PAIR*, ADJUST=*node_set_label*
secondary_surface, main_surface
...

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Adjust secondary nodes in set:** *node_set_label*

Adjusting Overclosed Secondary Nodes Using a Node Set Label

Because Abaqus/Standard adjusts only the secondary nodes in the specified node set, any overclosed secondary nodes not in the specified node set remain overclosed at the start of the simulation. Using a node set label may, therefore, cause convergence problems if severely overclosed secondary nodes, which need to be adjusted, are not included in the node set. This behavior is different from that seen if *a* is specified, in which case Abaqus/Standard adjusts all of the overclosed nodes on the secondary surface.

Comments Unique to Surface-to-Surface Contact

The following points apply to the surface-to-surface discretization (see *Contact Formulations in Abaqus/Standard* for further discussion of surface-to-surface discretization):

- Strain-free adjustments to secondary node positions may not result in exactly zero gap with respect to the main surface as measured at a secondary node. The adjustments are made to achieve zero gap between the surfaces in an average sense in a region near each secondary node within the adjustment zone.
- Strain-free adjustments will occur for some secondary nodes outside the adjustment zone if a significant portion of a secondary face (or segment in two dimensions) to which it is attached is within the adjustment zone.

Adjustments for Overlapping Contact Pairs

Nodal adjustment definitions are processed sequentially at the start of an analysis. If different constraint or contact definitions involve the same nodes, some adjustments may cause lack of compliance for contact or constraint definitions that were previously processed. These conflicts can be avoided in some cases by changing the processing order of constraint and contact definitions: nodes in common between different contact or constraint definitions should be processed first as secondary nodes and later as main nodes.

Input File Usage: To change the processing order of constraint and contact definitions, change the order of the definitions in the input file. Constraint and contact definitions are processed in the order in which they appear.

Abaqus/CAE Usage: To change the processing order of constraint and contact definitions, change the names of the constraints and interactions in the model. Constraints and interactions are processed alphabetically according to their name.

Storing Penetration Offsets for Noncompliant Surfaces to Be Initially in Contact

In some cases adjustments to the initial positions of surfaces to achieve zero initial penetration cause element distortion issues. It is generally recommended that you resolve the clash in the initial mesh by repositioning parts or making other changes to the model. However, for small-sliding and tied contact pairs, another modeling technique is available. You can specify that Abaqus/Standard introduce offset distance terms in the contact penetration calculations such that the modified initial penetration is zero even though the configuration appears to have an initial penetration or gap. This contact offset persists throughout the simulation and can cause confusion with respect to interpreting configurations.

This method involves explicitly setting the "adjustment zone" parameter to indicate that no initial surface position adjustments are to be made and specifying a position tolerance, *a*. Abaqus/Standard creates a penetration offset for secondary nodes within a gap distance, *a*, or overclosed with respect to the main surface.

Input File Usage: **CONTACT PAIR*, ADJUST=NO, POSITION TOLERANCE=*a*
secondary_surface, main_surface
...

Abaqus/CAE Usage: Storing penetration offsets in the contact penetration calculations is not supported in Abaqus/CAE.

Initial Clearance or Overclosure for Small-Sliding Contact

You can define precise initial clearance or overclosure values and contact directions for the nodes on the secondary surface when they would not be computed accurately enough from the nodal coordinates; for example, if the initial clearance is very small compared to the coordinate values. For more information, see *Additional Contact Initialization Options for Small-Sliding Contact in Abaqus/Standard*.

You can define initial clearance or overclosure values only for small-sliding contact (*Contact Formulations in Abaqus/Standard*). For a technique that can be used to model clearances or overclosures between finite-sliding contact pairs, see *Alternative Methods for Specifying Precise Initial Clearances or Overclosures* below.

Alternative Methods for Specifying Precise Initial Clearances or Overclosures

Abaqus/Standard offers an alternative method of defining precise initial clearances or overclosures that is applicable to both small-sliding and finite-sliding contact pairs. In this method you specify an adjustment zone depth for the contact pair (as described above in *Adjusting Initial Surface Positions to Resolve Small Initial Gaps or Overclosures*) to move the surfaces forming the contact pair exactly into contact at the start of the analysis. Then, in the first step of the simulation you specify an allowable contact interference, $\boldsymbol{v}$, for the contact pair (see *Modeling Contact Interference Fits in Abaqus/Standard*). The contact interference definition must refer to an amplitude curve; the form of the amplitude curve depends on whether a clearance or an overclosure is being defined and is described below. The clearance or overclosure will be uniform across the surfaces.

Input File Usage: Use all of the following options:

**CONTACT PAIR*, ADJUST=*a*
secondary_surface, main_surface
**AMPLITUDE*, NAME=*amplitude_name*
**CONTACT INTERFERENCE*, AMPLITUDE=*amplitude_name*
secondary_surface, main_surface, $\boldsymbol{v}$

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Specify tolerance for adjustment zone**: *a*, **Interference Fit**: toggle on **Uniform allowable interference**, **Amplitude**: *amplitude_name*, **Magnitude at start of step**: $\boldsymbol{v}$

Specifying a Precise Clearance by Defining an Allowable Contact Interference

To specify a precise clearance by defining an allowable contact interference, the amplitude curve should have a constant magnitude for the duration of the step. A positive value should be given as the allowable interference, $\boldsymbol{v}$. When viewed in Abaqus/CAE, these surfaces will appear to penetrate each other when they are in contact. The surfaces start the simulation with coordinates that have them exactly touching, but the specified interference $\boldsymbol{v}$ makes them behave as if they have a clearance between them.

Specifying a Precise Overclosure by Defining an Allowable Contact Interference

To specify a precise overclosure by defining an allowable contact interference, the amplitude curve should ramp from zero to unity over the duration of the step to allow Abaqus/Standard to resolve the overclosure gradually. A negative value should be given as the allowable interference, $\boldsymbol{v}$. When viewed in Abaqus/CAE, the surfaces start the simulation with coordinates that have them exactly touching, but the specified interference $\boldsymbol{v}$ makes them behave as if they are overclosed. As Abaqus/Standard resolves the overclosure, these surfaces will appear to separate from each other. When the gap between the two surfaces is equal to a distance of $|\boldsymbol{v}|$, the surfaces will behave as if they are precisely in contact.

Additional Contact Initialization Options for Small-Sliding Contact in Abaqus/Standard

Products: Abaqus/Standard

References:

- *About Contact Pairs in Abaqus/Standard*
- *About General Contact in Abaqus/Standard*

Overview

In Abaqus/Standard you can define precise initial clearance or overclosure values and contact directions for contact initialization in small-sliding contact for both contact pairs and general contact.

Defining Precise Initial Clearance or Overclosure Values

You can define precise initial clearance or overclosure distances and contact directions when they would not be computed accurately enough from the surface geometry. You can define initial clearance or overclosure values only for small-sliding contact (*Contact Formulations in Abaqus/Standard*) whether contact is modeled with contact pairs or with general contact. For a technique that can be used to model clearances or overclosures between finite-sliding contact pairs, see *Alternative Methods for Specifying Precise Initial Clearances or Overclosures*.

The initial clearance or overclosure value calculated for a potential contact constraint based on surface geometry is overwritten by the value that you specify. This procedure is performed internally, and it does not affect nodal positions. If you define positive clearance, Abaqus/Standard treats the two surfaces as not being in contact, regardless of their nodal coordinates. If you define an overclosure, Abaqus/Standard treats the two surfaces as an interference fit and attempts to resolve the overclosure in the first increment. If the defined overclosure is large, you might need to specify an allowable interference that is ramped off over several increments. See *Modeling Contact Interference Fits in Abaqus/Standard* for further discussion of interference fits.

Specifying a Uniform Clearance or Overclosure for the Surfaces

You can specify a uniform clearance or overclosure for a contact pair by identifying the main and secondary surfaces of the contact pair and the desired initial clearance, h_0 (positive for a clearance; negative for an overclosure). No other data are needed.

For general contact, you can specify a named clearance to associate with contact initialization for portions of the general contact domain where small sliding is active.

Input File Usage: Use the following option to specify a uniform clearance or overclosure for a contact pair:

**CLEARANCE*, SECONDARY=*surface_name*, MAIN=*surface_name*, VALUE=h_0

Use the following options to specify a uniform clearance or overclosure for general contact:

*CLEARANCE, NAME=*clrName*, VALUE=h_0
*CONTACT INITIALIZATION DATA, INITIAL CLEARANCE=*clrName*

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Clearance**: **Initial clearance: Uniform value across secondary surface: h_0**

Specifying initial clearance or overclosure for small-sliding contact in general contact is not supported in Abaqus/CAE.

Specifying Spatially Varying Clearances or Overclosures for the Surfaces

Alternatively, you can specify spatially varying clearances or overclosures by providing a table of data specifying nodal clearance values. You can specify these clearances for nodes of either surface of a small-sliding contact interaction within general contact; for small-sliding contact pairs, you must specify these clearances on nodes belonging to the secondary surface. General contact assigns implicit main-secondary roles by default. If you specify clearance values on nodes that act as main nodes, Abaqus/Standard interpolates the clearance used for an individual contact constraint from the clearance values of the main nodes participating in the contact constraint. If user-specified clearances do not apply to a particular contact constraint, Abaqus/Standard calculates the initial clearance from the initial geometry of the contacting surfaces.

For general contact, you can create a named clearance to associate with contact initialization for portions of the general contact domain where small sliding is active. For contact pairs, identify the main and secondary surfaces along with clearance data.

Input File Usage: Use the following option to specify spatially varying clearances or overclosures for contact pairs:

*CLEARANCE, SECONDARY=*surface_name*, MAIN=*surface_name*, TABULAR
node number or node set label, clearance value

Repeat the data line as often as necessary.

Use the following options to specify spatially varying clearances or overclosures for general contact:

*CLEARANCE, NAME=*clrName*, TABULAR
node number or node set label, clearance value

Repeat the data line as often as necessary.

*CONTACT INITIALIZATION DATA, INITIAL CLEARANCE=*clrName*, NAME=*C1*

The following options assign the surface *surf_1* a secondary role assuming the nodes with specified clearances belong to it before the surface pair *surf_1* and *surf_2* are assigned the named clearance during contact initialization.

*CONTACT
*CONTACT FORMULATION, TYPE=MAIN SECONDARY ROLES
surf_1, *surf_2*, SECONDARY
*CONTACT INITIALIZATION ASSIGNMENT
surf_1, *surf_2*, *C1*

Abaqus/CAE Usage: You cannot specify initial clearance or overclosure values using a table of data in Abaqus/CAE.

Reading Spatially Varying Clearances or Overclosures from an External File

Abaqus/Standard can read the spatially varying clearances or overclosures for a contact pair or general contact from an external file.

Input File Usage: Use the following option to specify spatially varying clearances or overclosures from an external file for contact pairs:

**CLEARANCE*, SECONDARY=*surface_name*, MAIN=*surface_name*, TABULAR, INPUT=*file_name*

Use the following options to specify spatially varying clearances or overclosures from an external file for general contact:

**CLEARANCE*, NAME=*clrName*, TABULAR, INPUT=*file_name*
**CONTACT INITIALIZATION DATA*, INITIAL CLEARANCE=*clrName*, NAME=*C1*

Abaqus/CAE Usage: You cannot specify initial clearance or overclosure values using an external input file in Abaqus/CAE.

Specifying the Surface Normal for the Contact Calculations

Normally Abaqus/Standard calculates the surface normal used for the contact calculations from the geometry of the discretized surfaces, using the algorithms described in *Contact Formulations in Abaqus/Standard*. When specifying spatially varying clearances or overclosures, you can redefine the contact direction that Abaqus/Standard uses with each secondary node by specifying the components of this vector. The vector must be defined in the global Cartesian coordinate system, and it should define the main surface's desired outward normal direction.

For general contact, instead of specifying the secondary and main roles, the clearance definition is identified by a name which is then associated with a contact initialization definition.

Input File Usage: Use the following option to specify the surface normal for the contact calculations for contact pairs:

**CLEARANCE*, SECONDARY=*surface_name*, MAIN=*surface_name*, TABULAR
node number or node set label, clearance value, first normal component, second normal component,
third normal component

Repeat the data line as often as necessary.

Use the following options to specify the surface normal for the contact calculations for general contact:

**CLEARANCE*, NAME=*clrName*, TABULAR, INPUT=*file_name*
**CONTACT INITIALIZATION DATA*, INITIAL CLEARANCE=*clrName*, NAME=*C1*

Abaqus/CAE Usage: You cannot redefine contact directions in Abaqus/CAE, except for threaded bolt connections (see *Generating Contact Normal Directions Based on a Reference Thread Geometry* below).

Generating Contact Normal Directions Based on a Reference Thread Geometry

This modeling approach provides a simple way to approximate the effects of threads without directly including threads in the mesh geometry. The meshed parts typically have cylindrical surfaces at the interface, such that the default contact normal directions are approximately radial. Abaqus/Standard adjusts the contact normal directions to be normal to the faces of the reference threads. *Figure 1* shows the planar approximation of the thread geometry where the modified contact normal direction is perpendicular to the plane.

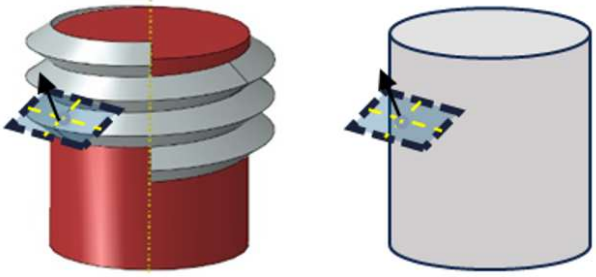


Figure 1: Local planar approximation of thread geometry associated with a small-sliding contact formulation.

The thread face normal directions have large components in the radial and axial directions. For three-dimensional models, the thread face normal directions also have a small component in the circumferential direction due to the spiral nature of the threads. Either the bolt or bolt hole can act as the secondary surface.

The capability to adjust contact normal directions based on reference thread geometry is available only for small-sliding contact formulations; therefore, it does not provide accurate results after relative twisting motion between a bolt and hole. For simulations involving relative twisting motion, you can consider the following alternative modeling approach:

1. Create nominal meshes without threads for the bolts and parts with bolt holes.
2. Create surface element meshes to capture the bolt thread geometry and bolt hole thread geometry.
3. Specify surface-based tie constraints to constrain each bolt thread surface to a bolt and each hole thread surface to a hole.
4. Specify finite-sliding contact (with penalty enforcement of contact constraints) between the respective thread surfaces.

For general contact, Abaqus/Standard automates Step 2 and Step 3 (see *Contact Surface Representation for Threaded Interfaces*).

Uniform Association with Top or Bottom Thread Face

By default with the capability based on reference thread geometry, all secondary nodes are assumed to consistently correspond to the "top" or "bottom" thread face, such that the adjusted contact normal directions for all secondary nodes have the same axial component and the same (small) circumferential component. In this case, the overall contact interface provides only "one-way" resistance to relative axial motion between the bolt and bolt hole.

You specify the thread geometry parameters along with two points "a" and "b," as shown in *Figure 2*, defining the axis of the bolt/bolt hole. The contact interface supports tension in a bolt if point "a" is near the bolt tip and point "b" is near the bolt head (as in *Figure 2*), and it supports compression in the bolt if points "a" and "b" have the opposite orientation. If a negative half-thread angle is specified, the opposite convention for supporting tension or compression occurs.

For general contact, instead of specifying the secondary and main roles, the clearance definition is identified by a name label to be associated with a contact initialization definition (as in earlier cases). Nodes specified for the

applicable region of the reference thread geometry can correspond to nodes of either surface of small-sliding contact interactions.

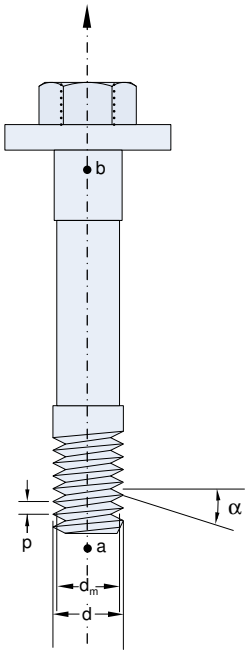


Figure 2: Reference thread geometry.

Input File Usage: Use the following option to specify a uniform association with the top or bottom thread face for contact pairs:

**CLEARANCE*, SECONDARY=*surface_name*, MAIN=*surface_name*, TABULAR, BOLT,
NORMAL ADJUSTMENT=UNIFORM AXIAL COMPONENT
half-thread angle, pitch, major bolt diameter, mean bolt diameter
node number or node set label, clearance value, coordinates of points a and b on the axis of the bolt/bolt hole

Repeat the second data line as often as necessary.

Use the following options to specify a uniform association with the top or bottom thread face for general contact:

**CLEARANCE*, NAME=*clrName*, TABULAR, BOLT, NORMAL
ADJUSTMENT=UNIFORM AXIAL COMPONENT
half-thread angle, pitch, major bolt diameter, mean bolt diameter
node number or node set label, clearance value, coordinates of points a and b on the axis of the bolt/bolt hole

Repeat the second data line as often as necessary.

**CONTACT INITIALIZATION DATA*, INITIAL CLEARANCE=*clrName*, NAME=*C1*

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Clearance**: **Initial clearance: Computed for single-threaded bolt** or **Specify for single-threaded bolt:** *clearance value*,
Secondary node set for clearance: Edit Region: select region,

Bolt direction vector: Edit: select axis,
Half-thread angle: *half-thread angle*, **Pitch:** *pitch*,
Bolt diameter: Major: *major bolt diameter* or **Mean:** *mean bolt diameter*
Normal adjustment: Uniform

Specifying initial clearance and/or contact directions for contact initialization for general contact is not supported in Abaqus/CAE.

Location-Dependent Association with Top or Bottom Thread Face

Optionally, each secondary node can be associated with the top or bottom thread face based on its initial position with respect to the reference thread geometry, which will influence the sign of the axial and circumferential components of the adjusted contact normal direction for each secondary node. Having some secondary nodes associated with the top thread face and others associated with the bottom thread face restricts motion of the bolt with respect to the hole in both axial directions and supports tension or compression in the bolt. It is unlikely that all secondary nodes will be associated with the same (top or bottom) thread face with this nondefault option. For example, consider that a purely circumferential mesh line would be associated with the top thread face over a 180° arc and the bottom thread face over the remaining 180° arc.

With this location-dependent association with top and bottom thread faces, contour plots of contact stress on bolt/hole surfaces often show stripes at intervals corresponding to the threads in the reference thread geometry.

Input File Usage: Use the following option to specify a location-dependent association with the top or bottom thread face for contact pairs:

**CLEARANCE*, BOLT, NORMAL ADJUSTMENT=LOCATION DEPENDENT

Use the following options to specify a location-dependent association with the top or bottom thread face for general contact:

**CLEARANCE*, NAME=*clrName*, BOLT, NORMAL ADJUSTMENT=LOCATION DEPENDENT
**CONTACT INITIALIZATION DATA*, INITIAL CLEARANCE=*clrName*, NAME=*C1*

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Clearance**: **Initial clearance: Computed for single-threaded bolt** or **Specify for single-threaded bolt:** *clearance value*, **Normal adjustment: Location dependent**

Specifying initial clearance and/or contact directions for contact initialization for general contact is not supported in Abaqus/CAE.

Right-Handed Versus Left-Handed Reference Thread Geometry

The reference thread geometry corresponds to right-handed threads by default; however, you can specify left-handed reference thread geometry optionally. Left-handed threads have the opposite effect on the circumferential component of the contact normal direction as right-handed threads.

Figure 3 shows that nodal contact force results on a bolt surface are not sensitive to whether thread geometry information is specified for nodes on one surface or the other for a small-sliding general contact interaction. In the case shown on the left, thread geometry information associated with left-handed threads is specified on the bolt (main) surface; in the case shown on the right, thread geometry information associated with left-handed threads is specified on the bolt-hole (secondary) surface. Abaqus/Standard internally assigns the bolt surface to act as the main surface for both of these simulations. For small-sliding contact pairs, you must specify thread

geometry information on the secondary surface. General contact automatically assigns main and secondary roles; in this example, the bolt acts as the main surface.

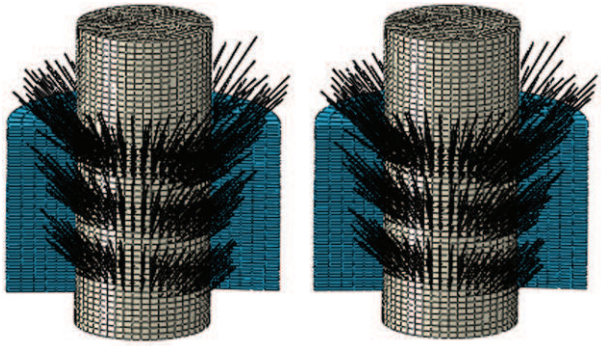


Figure 3: Cut-away views of contact force vectors.

Input File Usage: Use the following option to specify that the reference threads are right-handed for contact pairs:

**CLEARANCE*, BOLT, HANDEDNESS=RIGHT (default)

Use the following option to specify that the reference threads are left-handed for contact pairs:

**CLEARANCE*, BOLT, HANDEDNESS=LEFT

Use the following options to specify that the reference threads are right-handed for general contact:

**CLEARANCE*, NAME=*clrName*, BOLT, HANDEDNESS=RIGHT (default)
**CONTACT INITIALIZATION DATA*, INITIAL CLEARANCE=*clrName*, NAME=*C1*

Use the following options to specify that the reference threads are left-handed for general contact:

**CLEARANCE*, NAME=*clrName*, BOLT, HANDEDNESS=LEFT
**CONTACT INITIALIZATION DATA*, INITIAL CLEARANCE=*clrName*, NAME=*C1*

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Clearance**: **Initial clearance: Computed for single-threaded bolt** or **Specify for single-threaded bolt:** *clearance value*, **Handedness: Right** or **Left**

Specifying initial clearance and/or contact directions for contact initialization for general contact is not supported in Abaqus/CAE.

Visualizing the Precise Initial Clearances or Overclosures

Abaqus/Standard does not adjust the coordinates of the secondary surface when precise initial clearances or overclosures are specified. Therefore, the specified clearances or overclosures cannot be seen in the model in Abaqus/CAE. Thus, depending on the initial geometry of the surfaces and the magnitude of the clearances or overclosures, the surfaces might appear open or closed in Abaqus/CAE when they are actually just in contact. However, the actual clearance can be displayed in Abaqus/CAE by plotting a contour plot of the variable COPEN.

Contact Initialization for General Contact in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *About General Contact in Abaqus/Explicit*
- **CONTACT*
- **CONTACT INITIALIZATION ASSIGNMENT*
- **CONTACT INITIALIZATION DATA*
- **CONTACT CLEARANCE*
- **CONTACT CLEARANCE ASSIGNMENT*
- *Producing a deformed shape plot*
- *Specifying and modifying contact initialization assignments for general contact*

Overview

You can specify whether initial overclosures should be resolved without generating stresses, strains, or momentum or treated as interference fits that are gradually resolved over the first step.

In addition, contact initialization controls for general contact in Abaqus/Explicit:

- can be used to separate entangled double-sided surfaces;
- can be used to model an initial gap between surfaces;
- can influence which regions are initially bonded for cohesive contact;
- can be used to identify an initially bonded node set in crack propagation analyses;
- can be used to specify nondefault search zones that determine which nodes are affected by the contact initialization controls; and
- should not be specified to correct gross errors in the mesh design.

Defining and Assigning Contact Initialization Methods

You can define contact initialization methods that provide instructions for initializing contact, and then assign these methods to surface pairings. The default contact initialization method is in effect wherever you have not explicitly assigned a contact initialization method. For example, you may want to increase the tolerance for resolving initial deep penetrations or specify that certain gaps should be adjusted to a "just touching" status in some interface regions. Furthermore, some analyses call for initial overclosures to be treated as interference fits rather than resolved with strain-free adjustments.

Categories of characteristics associated with contact initialization methods include:

- The desired gap distance or interference-fit penetration distance for the beginning of the simulation (see *Controlling Initial Gap or Penetration State*).
- Whether nodes are repositioned or penetration offsets are stored to achieve the desired initial gap or interference-fit penetration state (see *Resolving Discrepancies between Mesh Geometry and Desired Initial Clearance or Penetration State*).
- Search criteria associated with application of the other contact initialization controls and identification of partially bonded surfaces for a VCCT crack propagation analysis (see *Contact Initialization Search Criteria*).

Two interfaces associated with contact initialization are available in Abaqus/Explicit. The preferred interface allows more general types of initialization. Usages for the preferred interface are included in the sections referenced above; usages for the alternative interface are described in *Alternative Contact Initialization User Interface*. Neither interface supports the use of edge-based surfaces, and contact edges will be automatically excluded from element-based surfaces as well.

You assign a name to each contact initialization method. This name is used in the assignment of a contact initialization method to specific surface pairings. Surface names used in the assignment of contact initialization methods to surface pairings do not have to correspond to the surface names used to specify the general contact domain. In many cases nondefault contact initialization methods are assigned to a subset of the overall general contact domain. Any contact initialization assignments for regions that fall outside of the general contact domain are ignored. The last assignment takes precedence if the specified interactions overlap. Special considerations for contact initialization assignments in the context of shell surfaces are discussed in *Contact Initialization Details Specific to Shells*.

Input File Usage: Use the following option to define a contact initialization method:

**CONTACT INITIALIZATION DATA*, NAME=*contact_initialization_method_name*

Use the following option to assign a contact initialization method to interactions:

**CONTACT INITIALIZATION ASSIGNMENT*
surface_1, surface_2, contact_initialization_method_name

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**: **Name:** *contact_initialization_method_name*

Interaction module: **Create Interaction**: **General contact (Explicit)**: **Contact Properties**: **Initialization assignments: Edit**: select the surfaces and the initialization in the columns on the left, and click the arrows in the middle to transfer them to the list of contact initialization assignments

Controlling Initial Gap or Penetration State

The desired initial contact state often differs from what can be interpreted from the specified mesh geometry. For example, regions with initial mesh overclosures are often intended to have surfaces treated as just touching with no initial overlap. Other regions with small gaps between parts according to the initial mesh may be intended to be overclosed by a distance associated with an interference fit. This section describes the default behavior and alternatives.

Default Initialization of Initial Gap or Penetration State

Abaqus/Explicit automatically adjusts the positions of surfaces or optionally stores penetration offsets (as discussed in *Resolving Discrepancies between Mesh Geometry and Desired Initial Clearance or Penetration State*) to remove small initial overclosures. In addition, if the search zone for contact initialization is extended to the gap region (as discussed in *Contact Initialization Search Criteria*), Abaqus/Explicit removes small gaps existing in the general contact domain in the first step of a simulation. Position adjustments are made with strain-free initial displacements. *Figure 1* shows an example of repositioning nodes to resolve initial overclosures. This automatic adjustment of the initial penetration or gap state is intended to correct only minor mismatches associated with mesh generation and is done even if the interaction is defined through user subroutine *VUINTERACTION*.

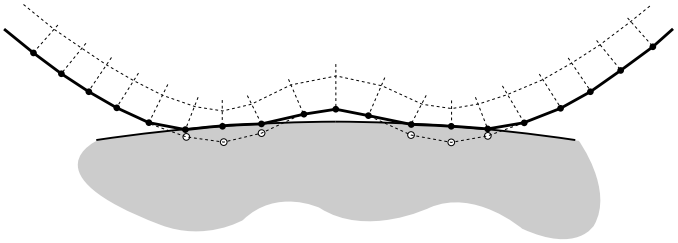


Figure 1: Configuration of contact surfaces after strain-free adjustments to resolve overclosure.

Specifying an Initial Clearance Distance

By default, adjustments to the initial penetration or gap state adjust initial nodal positions such that surfaces are "just touching" (with zero penetration/separation). Alternatively, you can specify a single non-negative value representing a uniform clearance or a nodal distribution to define a clearance per node (see *Distribution Definition*). If a distribution is defined and the clearance is omitted for a secondary node, the clearance value is interpolated from the values at the main nodes. The secondary node is ignored if clearance values are omitted from the distribution for the secondary node and all of the nodes of the nearest main face. The adjustments occur only for regions that satisfy the search zone tolerances, as discussed in *Contact Initialization Search Criteria*. Mesh distortion can occur if large strain-free adjustments occur to achieve the specified initial clearance distance. *Figure 2* shows an example in which a uniform initial gap is desired between concentric surfaces.

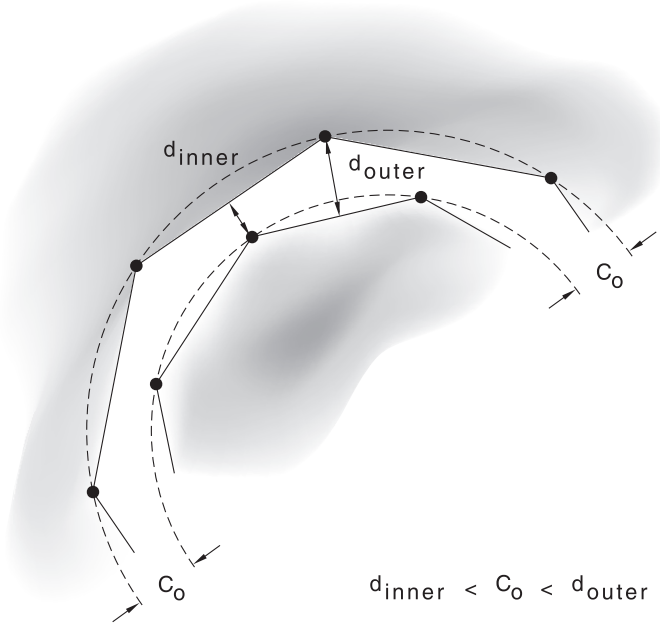


Figure 2: Specifying a uniform initial gap between concentric circular surfaces.

Input File Usage: **CONTACT INITIALIZATION DATA*, INITIAL CLEARANCE=*value* or *distribution_name*

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**; **Abaqus/Explicit contact initialization**; **Specify clearance distance:** *h*

Modeling Interference Fits

The general contact algorithm in Abaqus/Explicit can optionally treat initial overclosures as interference fits. In this case the interference distance usually corresponds to the original mesh overclosure distance although exceptions are discussed in *Specifying an Interference Distance*. Subsequent resolution of this interference is effectively a form of loading within the first step and typically generates stresses and strains.

The general contact algorithm uses a shrink-fit method to gradually resolve the interference distance over the first step of the analysis as shown in *Figure 3*. Stresses and strains are generated as the interference is resolved. Large velocities can develop if the interference fit is resolved too rapidly. You should follow these general recommendations:

- Do not apply other loads while the interference fit is being resolved.
- Resolve the interference fit over a large enough number of increments such that the kinetic energy remains small.

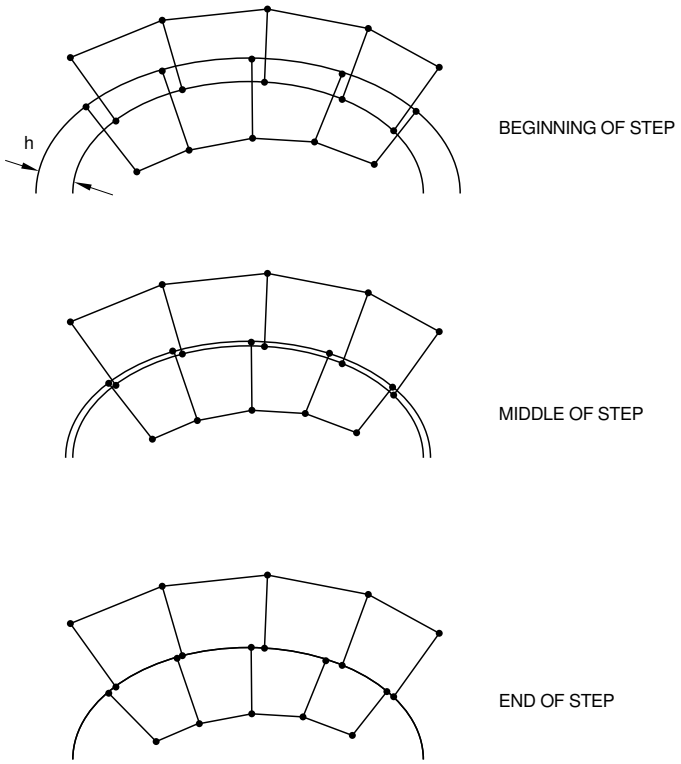


Figure 3: Gradual resolution of contact interference fit.

Input File Usage: **CONTACT INITIALIZATION DATA*, NAME=*Interference*, INTERFERENCE FIT

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**; **Abaqus/Explicit contact initialization**; **Treat as interference fits**

Specifying an Interference Distance

By default, when the interference fit method is specified, the interference distance is implied by the initial overclosure of the mesh; alternatively, you can specify a uniform interference distance. In this case Abaqus/Explicit first makes strain-free adjustments of nodal positions such that initial overclosure in the adjusted configuration

corresponds to the specified interference distance and then invokes the shrink-fit method discussed above, as depicted in *Figure 4*. Mesh distortion can occur if large strain-free adjustments are necessary to achieve the specified interference distance.

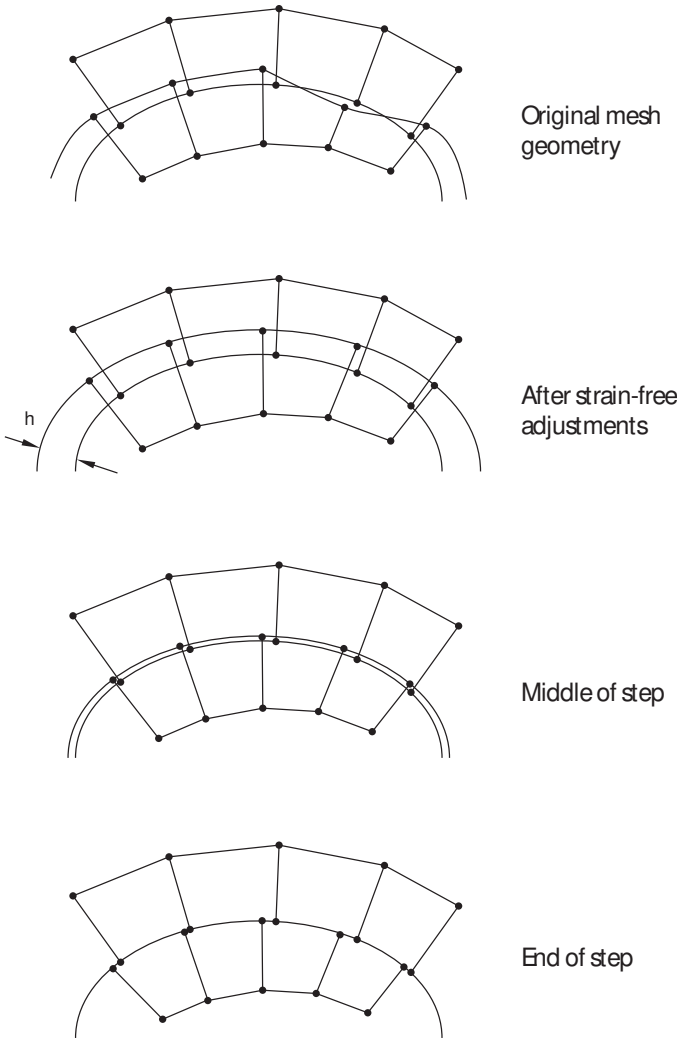


Figure 4: Treatment of a specified interference distance that differs from the interference implied by the original mesh.

Input File Usage: **CONTACT INITIALIZATION DATA*, NAME=*Interf_Dist*, INTERFERENCE FIT=*h*

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**; **Abaqus/Explicit contact initialization**; **Treat as interference fits**: **Specify interference distance:** *h*

Resolving Interference Fit over a Fraction of a Step Duration

Optionally, you can specify a fraction of the step time over which the interference fit is resolved. For example, if the interference fit is resolved during the first three-quarters of the step, some damping of dynamic effects occurs without additional loading associated with resolving the interference over the last quarter of the step. While this fraction does not affect contact initialization, it is specified within the contact initialization definition for convenience.

Input File Usage: **CONTACT INITIALIZATION DATA*, NAME=*Interf_StepFrac*, INTERFERENCE FIT, STEP FRACTION=*f*

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**; **Abaqus/Explicit contact initialization**; **Treat as interference fits**:**Step fraction for interference:** *f*

Resolving Discrepancies between Mesh Geometry and Desired Initial Clearance or Penetration State

Controlling Initial Gap or Penetration State discusses how to specify an initial gap or penetration state that may differ from that implied by the specified mesh geometry. This section discusses options available in Abaqus/Explicit to resolve descrepancies between the prescribed state and the specified mesh geometry.

Achieving Desired Initial Clearance or Penetration State with Strain-Free Adjustments

By default, Abaqus/Explicit automatically adjusts positions of surfaces to resolve discrepancies between gap or overclosure of the original mesh geometry and the desired initial gap or penetration state among general contact interactions of the first step. These adjustments are made with strain-free initial displacements to secondary node positions. This automatic adjustment of nodal positions is intended to correct only minor mismatches associated with mesh generation.

Output variable STRAINFREE (see *Abaqus/Explicit Output Variable Identifiers*) can be used to review strain-free adjustments. This output variable contains nodal vectors representing initial strain-free adjustments. By default, STRAINFREE is written to the output database (`.odb`) file for the original field output frame at zero time if any strain-free adjustments are made by Abaqus/Explicit. A symbol plot of this variable in the Visualization module of Abaqus/CAE shows vectors that represent how individual nodes have been adjusted, and a contour plot of this variable shows the distribution of the adjustment magnitude. You must select the original output frame at zero time in the Visualization module of Abaqus/CAE before choosing the STRAINFREE output variable.

Input File Usage: Use the following to resolve discrepancies by adjusting nodal coordinates:

**CONTACT INITIALIZATION DATA*, NAME=*contact_initialization_method_name*, ADJUST=YES

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**; **Abaqus/Explicit contact initialization**; **Adjust nodal coordinates**

Limiting Strain-Free Adjustments to One Surface in a Contact Initialization Assignment

Most interactions in Abaqus/Explicit are treated with a balanced main-secondary approach, such that the surfaces of an interaction act as both secondary and main (see *Contact Surface Weighting*). In this case, nodal position adjustments occur for both surfaces of an interaction. Optionally, you can limit strain-free nodal position adjustments to one surface of a contact initialization assignment even in such cases.

Input File Usage: Use the following option to limit strain-free adjustments to the first surface of a contact initialization assignment:

**CONTACT INITIALIZATION ASSIGNMENT*
surface_1, surface_2, contact_initialization_method_name, FIRST SURFACE SFC

Use the following option to limit strain-free adjustments to the second surface of a contact initialization assignment:

**CONTACT INITIALIZATION ASSIGNMENT*
surface_1, surface_2, contact_initialization_method_name, SECOND SURFACE SFC

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Contact Properties**: **Initialization assignments: Edit**: select the surfaces, the initialization in the columns and secondary surface option type on the left, and click the arrows in the middle to transfer them to the list of contact initialization assignments

Achieving a Desired Initial Zero or Positive Clearance State with Contact Offsets Instead of Strain-Free Adjustments

Discrepancies between gap or overclosure of the original mesh geometry and a desired initial clearance state specified can optionally be resolved with contact offsets instead of strain-free adjustments. This option is not applicable to contact initialization methods involving interference fit. Contact offsets effectively modify penetration calculations for individual secondary nodes, such that the contact force is computed as $f = k\left(d_{cur} - d_{offset}\right)$; where k is the penalty stiffness, d_{offset} is the contact offset distance, and d_{cur} is the current penetration distance. The contact offset distance remains constant for a given secondary node if you deactivate strain-free adjustments.

The use of contact offsets instead of strain-free adjustments (or repositioning parts in a preprocessor to avoid overclosures) can lead to confusion and nonphysical behavior for the following reasons:

- The contact force is proportional to the modified penetration $d_{mod} = d_{cur} - d_{offset}$, and this modified penetration differs from the penetration inferred by visual inspection of the current configuration of the model.
- The use of offsets, especially large offsets, tends to degrade robustness of the contact algorithms. You can request initial overclosure and crossed surface diagnostics to diagnose some of these problems (see *Contact Diagnostics in an Abaqus/Explicit Analysis*).

Input File Usage: Use the following option to deactivate strain-free adjustments:

**CONTACT INITIALIZATION DATA*, NAME=*contact_initialization_method_name*, ADJUST=NO

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**; **Abaqus/Explicit contact initialization**; **Adjust nodal coordinates**

Other Cases Involving Contact Offsets

Contact offsets are used by default to treat discrepancies between the original mesh geometry and the desired initial contact state in the following cases:

- Discrepancies remaining after strain-free adjustments, which may be due to conflicting adjustments from separate contact (such as when a secondary node that is part of a shell is pinched between two main facets), boundary conditions, tie constraints, coupling constraints, and rigid body constraints.
- Discrepancies associated with contact interactions newly included in general contact after the first step if no contact was defined in the previous step.

- Discrepancies associated with contact interactions for an Abaqus/Standard analysis imported into Abaqus/Explicit if the contact interaction is not defined with user subroutine *VUINTERACTION*.

In these cases, contact force is computed with $f = k(d_{cur} - d_{offset})$, as discussed in *Achieving a Desired Initial Zero or Positive Clearance State with Contact Offsets Instead of Strain-Free Adjustments*, but the contact offset distance, d_{offset}, is reduced as surfaces separate. If d_{cur} ever decreases below d_{offset}, d_{offset} is reset to d_{cur}.

Cases with Undesired Initial Overclosures That Are Not Resolved

If the general contact domain is extended after the first step, Abaqus/Explicit takes no special actions, as part of initialization for the new step, to adjust nodal positions, store penetration offsets, or gradually resolve initial penetrations for the newly introduced interactions. Penalty contact forces are applied proportional to the current penetration, or the penetration may be ignored. Furthermore, contact initialization diagnostics are not available for these new interactions.

Contact Initialization Search Criteria

You can specify search distances to define search "zones" above and below the surfaces. Other aspects of the assigned contact initialization method are applied to secondary nodes that lie within these search zones. By default, the search zones exclude:

- Initial overclosures into a solid surface greater than any contact thickness in effect plus the larger of approximately 15% of the dimension of facets attached to the secondary node and a specified interference fit distance.
- Initial overclosures into a shell surface greater than the contact thickness in effect, except for situations discussed in *Contact Initialization Details Specific to Shells* for which a single-sided shell surface is used in the contact initialization assignment.
- Initial gaps greater than the larger of a tiny fraction of the dimension of facets and a specified initial clearance distance.

For example, *Figure 5* shows two solid elements with large initial penetrations that will not be detected during the default initial overclosure resolution procedure. Potential interactions associated with the excluded initial overclosures are not eligible for contact until validation criteria are met for newly established contact (such as if the two parts separated and then came back into contact during the simulation).

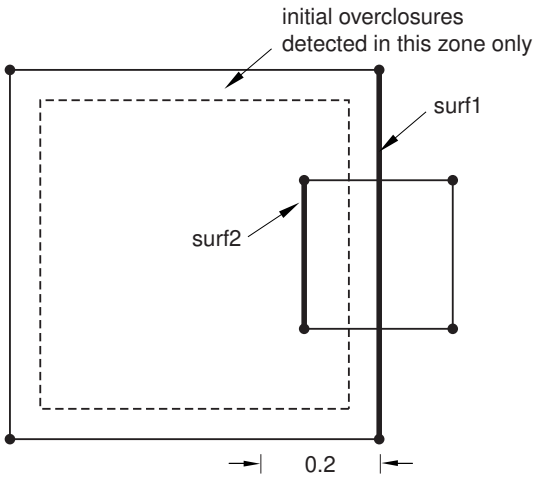


Figure 5: Undetected large penetrations of solid elements.

You can increase the overclosure threshold of the initialization search criteria by explicitly specifying a "search below" distance. If you specify a search below distance smaller than the default overclosure tolerance and assign this contact initialization method to interactions between surf1 and surf2, Abaqus/Explicit uses the default tolerance instead. If the default desired initial clearance of zero is in effect, increasing the search below distance to 0.2 causes the initial overclosures to be resolved as shown in *Figure 6*.

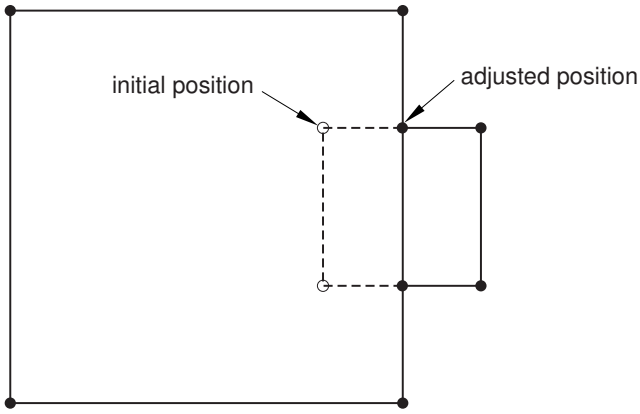


Figure 6: Resolution of large penetrations of solid elements.

Adjusting the nodal coordinates can do the following:

- Degrade the mesh geometry by creating imperfections that were not initially present.
- Reduce the element size and correspondingly the stable time increment size.
- Cause elements to invert and prevent the analysis from continuing.

In such cases it may be preferable to bypass the nodal coordinate adjustments and specify the storage of a contact offset (as discussed in *Resolving Discrepancies between Mesh Geometry and Desired Initial Clearance or Penetration State*) or to modify part positioning in the preprocessor to avoid overclosures.

You can close small initial gaps between surfaces by specifying a "search above" distance.

Increasing the extent of the search zones for strain-free adjustments can potentially increase the computational cost of an analysis. It is not generally recommended that you specify a large search zone because this may cause mesh distortion when nodes are repositioned over large distances.

Input File Usage: **CONTACT INITIALIZATION DATA*, NAME=*contact_initialization_method_name*, SEARCH ABOVE=*a*, SEARCH BELOW=*b*

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**: **Name:** *contact_initialization_method_name*; **Abaqus/Explicit contact initialization**; **Resolve with strain-free adjustments**: **Ignore overclosures greater than:** *b*, **Ignore initial openings greater than:** *a*

Using a Node Set as the Search Criterion or to Identify Initially Bonded Nodes

As an alternative to specifying search distances, you can specify a search node set containing the secondary nodes for which nondefault contact initialization instructions are meant to apply. However, the nondefault contact initialization instructions apply only for nodes of this set within a fixed distance from the main surface. The fixed distance is the maximum element size for solid elements or the thickness for structural elements (for example, shell elements) associated with the nodes.

You can also specify a search node set to identify which secondary nodes will be tagged as initially bonded in a VCCT crack propagation analysis. See *Crack Propagation Analysis* for more details.

Input File Usage: **CONTACT INITIALIZATION DATA*, NAME=*contact_initialization_method_name*, SEARCH NSET=*node set name*

Abaqus/CAE Usage: Interaction module: **Interaction->Contact Initialization->Create**: **Name:** *contact_initialization_method_name*; **Abaqus/Explicit contact initialization**; **Resolve with strain-free adjustments**: **Secondary node set for clearance:** *node set name*

Contact Initialization Details Specific to Shells

Contact initialization involving parts modeled with shells may require extra attention due to the lack of a unique outward direction. For example, *Figure 7* shows two shells separated by some distance in the initial mesh geometry, along with two possible adjustments to initially touching (zero gap) configurations. The case shown on the right preserves Surface 1 above Surface 2 before and after the nodal position adjustments, whereas the case shown on the left has Surface 2 below Surface 1 after nodal position adjustments. These cases differ in the following initialization settings to achieve the respective adjustments:

- Left side: "Search below" tolerance set to at least as large as the overclosure distance shown on the top left of *Figure 7*. Single-sided shell surfaces (created with SPOS or SNEG side identifiers explicitly specified) corresponding to the top of Surface 1 and the bottom of Surface 2 are used in the contact initialization assignment. Single-sided surfaces used in contact initialization assignments must have consistent surface normal orientations for adjacent faces.
- Right side: "Search above" tolerance set to at least as large as the gap distance between the shells in the initial mesh configuration, as discussed in *Contact Initialization Search Criteria*. Double-sided shell surfaces (created without side identifiers specified) can be used for Surface 1 and Surface 2 in the contact initialization assignment because the initialization algorithm by default preserves Surface 1 remaining on the top side of Surface 2. Alternatively, using single-sided surfaces corresponding to the bottom of Surface 1 and the top of Surface 2 also leads to the adjusted configuration shown on the bottom right of *Figure 7*.

Both of these cases call for retaining default settings of the desired initial gap distance (zero) and the method (nodal position adjustments) used to achieve the desired initial gap distance.

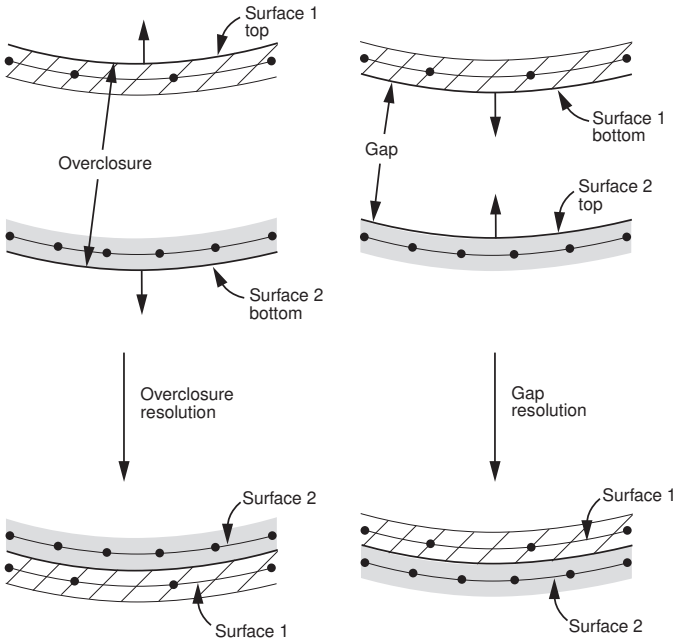


Figure 7: Two interpretations of adjusting nearby shells to "just touching."

Figure 8 shows a side view of a case with secondary nodes on different sides of a shell main reference surface. Four nodes of the original mesh geometry are initially within the region spanned by the shell thickness. The figure also shows the default corrected positions of secondary nodes to achieve zero initial clearance at secondary node locations while keeping each secondary node on the same side of the reference surface as in the original mesh geometry.

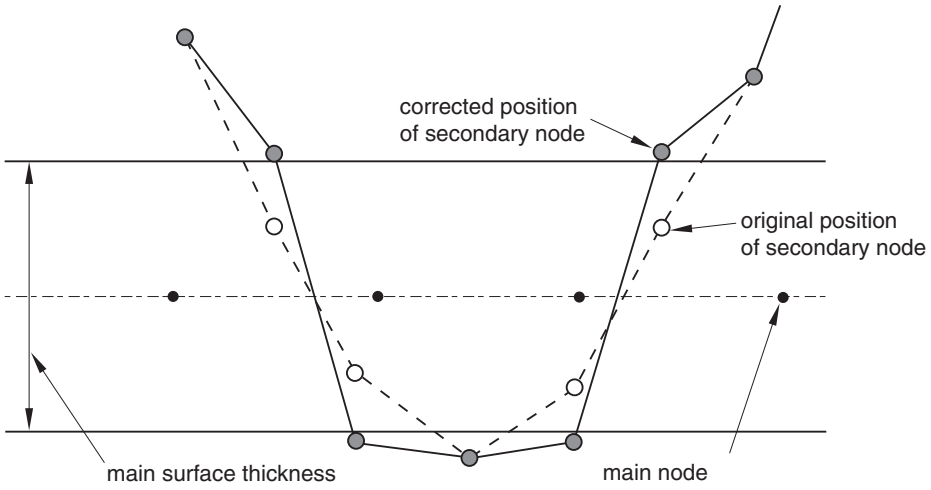


Figure 8: Default resolution of initial overclosure for tangled surfaces.

Secondary nodes that are trapped on opposite sides of a double-sided main surface often lead to serious problems involving nonphysical behavior during a simulation. A secondary surface initially crossing the main shell reference surface may or may not involve some initially overclosed nodes in the original mesh geometry, so the STRAINFREE output variable may or may not bring attention to these regions. Warning messages are provided for cases with adjacent secondary nodes trapped on opposite sides of a double-sided main surface. Similar to the discussion of *Figure 7*, the use of single-sided surfaces (created with SPOS or SNEG side identifiers) in contact initialization assignments provides a mechanism to detangle surfaces that are crossed in the initial mesh geometry. For example, the use of single-sided surfaces in the contact initialization assignment along with an increase in the "search below" distance leads to all secondary nodes adjusted to the top side of the main surface, as shown in *Figure 9*.

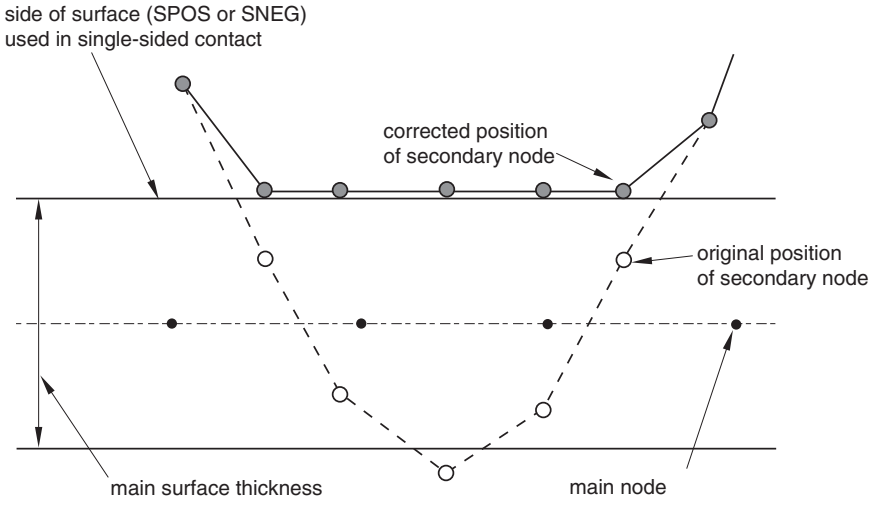


Figure 9: Improved resolution of initial overclosure for tangled surfaces.

Alternative Contact Initialization User Interface

The alternative contact initialization user interface applies to a subset of functionality of the preferred contact initialization user interface and focuses on specifying contact clearances. The alternative contact initialization user interface cannot be used to specify contact interference. With the alternative method, you specify contact clearance methods and assign contact clearance methods to interactions. Only single-sided surfaces can be used in the assignment of contact clearance methods, whereas single- or double-sided surfaces can be used for assignments with the preferred user interface.

Input File Usage: Use the following option to define a contact clearance method:

**CONTACT CLEARANCE*, NAME=*contact_clearance_method_name*

Use the following option to assign a contact clearance method to interactions:

**CONTACT CLEARANCE ASSIGNMENT*
surface_1, surface_2, contact_clearance_method_name

Abaqus/CAE Usage: The alternative contact initialization user interface is not supported in Abaqus/CAE.

Specifying an Initial Clearance Distance (Alternative Interface)

In general, the concepts of specifying an initial clearance distance with the alternative interface are the same as with the preferred interface (see *Specifying an Initial Clearance Distance*); however, the syntax for specifying the initial clearance distance is different.

Input File Usage: **CONTACT CLEARANCE*, NAME=*contact_clearance_method_name*, CLEARANCE=*value* or *distribution_name*

Abaqus/CAE Usage: The alternative contact initialization user interface is not supported in Abaqus/CAE.

Achieving Desired Initial Clearance with Strain-Free Adjustments (Alternative Interface)

In general, the concepts of achieving an initial clearance distance with strain-free adjustments using the alternative interface are the same as with the preferred interface (see *Achieving Desired Initial Clearance or Penetration State with Strain-Free Adjustments*); however, the syntax for specifying this method is different.

Input File Usage: **CONTACT CLEARANCE*, NAME=*contact_clearance_method_name*, ADJUST=YES

Abaqus/CAE Usage: The alternative contact initialization user interface is not supported in Abaqus/CAE.

Limiting Strain-Free Adjustments to One Surface (Alternative Interface)

In general, the concepts of limiting strain-free adjustments to one surface using the alternative interface are the same as with the preferred interface (see *Limiting Strain-Free Adjustments to One Surface in a Contact Initialization Assignment*); however, the syntax for specifying this method is different.

Input File Usage: Use the following option to limit strain-free adjustments to the first surface of a contact clearance assignment:

**CONTACT CLEARANCE ASSIGNMENT*
surface_1, surface_2, contact_clearance_method_name, SECONDARY

Use the following option to limit strain-free adjustments to the second surface of a contact clearance assignment:

**CONTACT CLEARANCE ASSIGNMENT*
surface_1, surface_2, contact_clearance_method_name, MAIN

Abaqus/CAE Usage: The alternative contact initialization user interface is not supported in Abaqus/CAE.

Achieving Desired Initial Clearance with Offsets (Alternative Interface)

In general, the concepts of achieving an initial clearance distance with offsets using the alternative interface are the same as with the preferred interface (see *Achieving a Desired Initial Zero or Positive Clearance State with Contact Offsets Instead of Strain-Free Adjustments*); however, the syntax for specifying this method is different.

Input File Usage: **CONTACT CLEARANCE*, NAME=*contact_clearance_method_name*, ADJUST=NO

Abaqus/CAE Usage: The alternative contact initialization user interface is not supported in Abaqus/CAE.

Contact Initialization Search Criteria (Alternative Interface)

In general, the concepts of modifying contact initialization search criteria using the alternative interface are the same as with the preferred interface (see *Contact Initialization Search Criteria*); however, the syntax for specifying this method is different.

Input File Usage: **CONTACT CLEARANCE*, NAME=*contact_clearance_method_name*, SEARCH ABOVE=*a*, SEARCH BELOW=*b*

Abaqus/CAE Usage: The alternative contact initialization user interface is not supported in Abaqus/CAE.

Using a Node Set as the Search Criterion or to Identify Initially Bonded Nodes (Alternative Interface)

In general, the concepts of using a node set as the search criteria or to identify initially bonded nodes using the alternative interface are the same as with the preferred interface (see *Using a Node Set as the Search Criterion or to Identify Initially Bonded Nodes*); however, the syntax for specifying this method is different.

Input File Usage: **CONTACT CLEARANCE*, NAME=*contact_clearance_method_name*, SEARCH NSET=*node set name*

Abaqus/CAE Usage: The alternative contact initialization user interface is not supported in Abaqus/CAE.

Contact Initialization for Contact Pairs in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *About Contact Pairs in Abaqus/Explicit*
- **CLEARANCE*
- **CONTACT PAIR*
- *Defining surface-to-surface contact*

Overview

Adjustments to the positions of the secondary nodes in an Abaqus/Explicit contact pair are performed for all contact pairs that have secondary nodes that are overclosed and that do not have specified initial clearances, except when nodes of a rigid body act as secondary nodes.

In addition, adjustments to the positions of the secondary nodes:

- are performed for all contact pairs that have secondary nodes that are overclosed and that do not have specified initial clearances, except when nodes of a rigid body act as secondary nodes;
- can eliminate small gaps or penetrations caused by numerical roundoff when a graphical preprocessor such as Abaqus/CAE is used;
- do not create any strains or momentum in the model during the first step of a simulation;
- do create strains and momentum in subsequent steps of a simulation;
- should not be used to correct gross errors in the mesh design; and
- should not be used to resolve initial overclosures involving a secondary node that is pinched between two main surfaces.

If the small-sliding contact formulation (see *Contact Formulations for Contact Pairs in Abaqus/Explicit*) is used, an alternative to adjusting the position of the surfaces is to define the initial clearances between the surfaces precisely in both magnitude and direction.

Adjustments of Overclosed Surfaces in the First Step of the Simulation

Abaqus/Explicit will automatically adjust the positions of surfaces to remove any initial overclosures that exist when a contact pair is defined in the first step of a simulation, except when nodes of a rigid body act as a secondary nodes or user subroutine *VUINTER* is used. The adjustments are made with strain-free initial displacements to the secondary nodes on the surfaces. Therefore, when a balanced main-secondary contact pair is defined, nodes on both surfaces may be adjusted. This automatic adjustment of surface position is intended to correct only minor mismatches associated with mesh generation. You can review the surface adjustments in the status (`.sta`) file, the message (`.msg`) file, and the output database (`.odb`) file; see *Contact Diagnostics in an Abaqus/Explicit Analysis* for more information.

Some softened contact models have nonzero contact pressure at zero overclosure (see *Contact Pressure-Overclosure Relationships*). For these models some initial, nonequilibrated contact pressure may be present at the beginning of an analysis, as the adjustments are made to satisfy zero overclosure rather than zero contact pressure. Large initial contact pressures may cause excessive distortion of elements near the contact surfaces.

Conflicting adjustments from separate contact pairs will cause incomplete resolution of initial overclosures and will lead to a noisy solution or severe distortion of elements. This can occur when a secondary node is pinched between two main surfaces.

Because of the lack of a unique outward direction from double-sided facets, the resolution of large initial penetrations for double-sided surfaces can be difficult. Initial penetration will be detected only when a secondary node lies within the thickness of the underlying element, and the initial penetration will be resolved by moving the secondary node to the nearest free surface as shown in *Figure 1*.

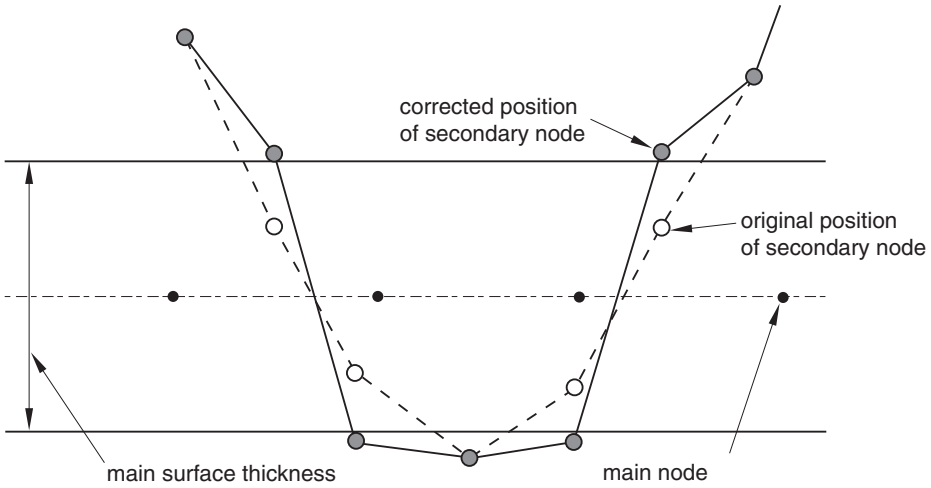


Figure 1: Correction of initial overclosure for a contact pair involving two double-sided surfaces.

A warning message will be issued to the status (`.sta`) file if two adjacent secondary nodes (connected by a facet edge) are detected on opposite sides of a double-sided main surface involved in contact defined with the contact pair algorithm. No such warning will be issued for node-based surface nodes on opposite sides of a double-sided main surface, because adjacency cannot be determined among the node-based surface nodes. If the main surface is a single-sided surface, initial overclosures will be resolved using the surface normal of the main surface, as shown in *Figure 2*.

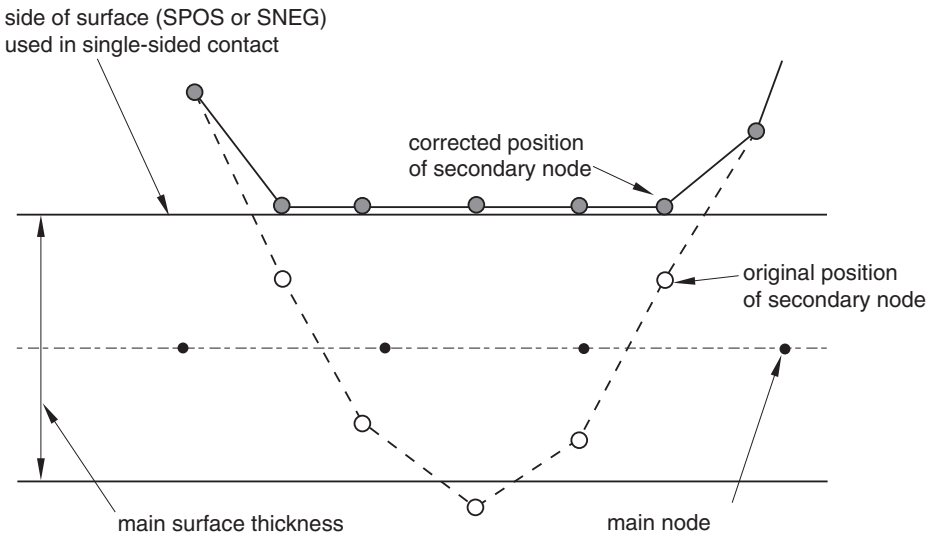


Figure 2: Correction of initial overclosure for a contact pair involving a single-sided and a double-sided surface.

Having secondary nodes trapped on opposite sides of a double-sided main surface will often lead to serious problems, which may not became apparent until later in an analysis. Therefore, a data check analysis (see

Abaqus/Standard and Abaqus/Explicit Execution) is recommended prior to running a large contact pair analysis so that you can check for warning messages in the status file (`.sta`) and check for mislocated adjacent secondary nodes on opposite sides of the main surface.

The adjustments affect only the nodes on the surfaces. Excessive distortion of neighboring elements may result if this feature is used to correct for gross errors in the initial geometry, causing the analysis to end with an error message.

Nodes on a rigid body can act as secondary nodes only for penalty contact pairs. Initial penetrations of secondary nodes that are part of a rigid body are not resolved with strain-free corrections; i.e., the secondary nodes are not adjusted. These penetrations are likely to cause artificially large contact forces in the first increments of an analysis and should, therefore, be avoided in the mesh definition.

Adjustments of Overclosed Surfaces during Subsequent Steps in the Simulation

If contact pairs are defined in later steps with initially overclosed surfaces, Abaqus/Explicit does not take any special actions to gradually resolve these initial penetrations: contact forces will be applied according to whatever contact constraint enforcement method is being used. These contact forces may be very large, causing large accelerations and velocities and possible distortion of elements. Initial penetrations have the potential to cause problems for contact pairs introduced in any step if a *VUINTER* user subroutine is used; but in that case you control the application of contact forces.

Minimizing the Noise Associated with Adjustments of Initially Overclosed Surfaces

When a balanced main-secondary contact pair is used for situations where the initial overclosure adjustments are not very small, non-negligible errors may persist in the adjusted geometry and can lead to a noisy oscillation (or “ringing”) in the contact procedure. This problem can sometimes be mitigated by modifying the contact pair to be a pure main-secondary relationship using a weighting factor; see *Contact Surface Weighting* for details.

Specifying Initial Clearance Values Precisely

You can define initial clearances and contact directions precisely for the nodes on the secondary surface when they would not be computed accurately enough from the nodal coordinates; for example, if the initial clearance is very small compared to the coordinate values. Initial clearances and contact directions can be defined only in small-sliding contact analyses (*Contact Formulations for Contact Pairs in Abaqus/Explicit*).

The initial clearance value calculated at every secondary node based on the coordinates of the secondary node and the main surface is overwritten by the value that you specify. This procedure does not alter the coordinates of the secondary nodes.

When the balanced main-secondary contact algorithm is invoked for the contact pair, the initial clearance values can be defined on one or both of the surfaces. Initial clearances defined on contact surfaces that act only as main surfaces will be ignored.

Specifying a Uniform Clearance for the Surfaces

You can specify a uniform clearance for a contact pair by identifying the contact pair and the desired initial clearance, h_0 (the value must be positive). No other data are needed.

Input File Usage: **CLEARANCE*, CPSET=*cpset_name*, VALUE=h_0

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Clearance**: **Initial clearance: Uniform value across secondary surface:** h_0

Specifying Spatially Varying Clearances for the Surfaces

Alternatively, you can specify spatially varying clearances for a contact pair by identifying the contact pair and a table of data specifying the clearance at a single node or a set of nodes belonging to the secondary surface. Any secondary surface node that is not identified will use the clearance that Abaqus/Explicit calculates from the initial geometry of the surfaces.

Input File Usage: **CLEARANCE*, CPSET=*cpset_name*, TABULAR

Abaqus/CAE Usage: You cannot specify initial clearance or overclosure values using a table of data in Abaqus/CAE.

Input File Usage: **CLEARANCE*, CPSET=*cpset_name*
node_set_label, *value*
node_number, *value*
...

Reading Spatially Varying Clearances from an External File

Abaqus/Explicit can read the spatially varying clearances for a contact pair from an external file.

Input File Usage: **CLEARANCE*, CPSET=*cpset_name*, TABULAR, INPUT=*file_name*

Abaqus/CAE Usage: You cannot specify initial clearance or overclosure values using an external input file in Abaqus/CAE.

Specifying the Surface Normal for the Contact Calculations

Normally Abaqus/Explicit calculates the surface normal used for the contact calculations from the geometry of the discretized surfaces, using the algorithms described in *Contact Formulations for Contact Pairs in Abaqus/Explicit*. When specifying spatially varying clearances, you can redefine the contact direction that Abaqus/Explicit uses with each secondary node by specifying the components of this vector. The vector must define the global Cartesian components of the outward normal to the main surface.

Input File Usage: **CLEARANCE*, SECONDARY=*surface_name*, MAIN=*surface_name*, TABULAR
node number or node set label, clearance value, first normal component, second normal component, third normal component

Repeat the data line as often as necessary.

Abaqus/CAE Usage: You cannot redefine contact directions in Abaqus/CAE, except for thread bolt connections (see *Generating the Contact Normal Directions for a Thread Bolt Connection Automatically* below).

Generating the Contact Normal Directions for a Thread Bolt Connection Automatically

Alternatively, for a single-threaded bolt connection the contact normal directions for each secondary node can be generated automatically by specifying the thread geometry data and two points used to define a vector on the

axis of the bolt/bolt hole. The axis vector should be oriented to point from the tip of the bolt to the head of the bolt when in tension and from the head to the tip when in compression.

Input File Usage: **CLEARANCE*, CPSET=*cpset_name*, TABULAR, BOLT
half-thread angle, pitch, major bolt diameter, mean bolt diameter
node number or node set label, clearance value, coordinates of points a and b on the axis of the bolt/bolt hole

Repeat the second data line as often as necessary.

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Clearance**: **Initial clearance: Computed for single-threaded bolt** or **Specify for single-threaded bolt:** *clearance value*,
Clearance region on secondary surface: Edit Region: select region,
Bolt direction vector: Edit: select axis,
Half-thread angle: *half-thread angle*, **Pitch:** *pitch*,
Bolt diameter: Major: *major bolt diameter* or **Mean:** *mean bolt diameter*

Contact Formulations and Numerical Methods

In this section:

- *Contact Formulations, Numerical Methods, and Controls in Abaqus/Standard*
- *Contact Formulations, Numerical Methods, and Controls in Abaqus/Explicit*

Contact Formulations, Numerical Methods, and Controls in Abaqus/Standard

In this section:

- *Contact Formulations in Abaqus/Standard*
- *Contact Constraint Enforcement Methods in Abaqus/Standard*
- *Smoothing Contact Surfaces in Abaqus/Standard*
- *Initial Contact Stresses in Abaqus/Standard*
- *Generally Applicable Contact Controls in Abaqus/Standard*
- *Contact Controls Specific to General Contact in Abaqus/Standard*

Contact Formulations in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Surfaces*
- *About General Contact in Abaqus/Standard*
- *About Contact Pairs in Abaqus/Standard*
- **CONTACT*
- **CONTACT PAIR*
- *Defining general contact*
- *Defining surface-to-surface contact*
- *Defining self-contact*
- *Using contact and constraint detection*

Overview

Abaqus/Standard provides several contact fomulations. Each formulation is based on a choice of a contact discretization, a tracking approach, and assignment of “main” and “secondary” roles to the contact surfaces. For general contact interactions, the discretization, tracking approach, and surface role assignments are selected automatically by Abaqus/Standard; for contact pairs, you can specify these aspects of the contact formulation using the interface described in *About Contact Pairs in Abaqus/Standard*. The default contact formulation is applicable in most situations, but you may find it desirable to choose another formulation in some cases. This section discusses in detail the formulations that Abaqus/Standard uses in contact simulations.

Your choice of a tracking approach will have a considerable impact on how contact surfaces interact. In Abaqus/Standard there are two tracking approaches to account for the relative motion of two interacting surfaces in mechanical contact simulations:

- finite sliding, which is the most general and allows any arbitrary motion of the surfaces (see *Finite-sliding interaction between deformable bodies* and *Finite-sliding interaction between a deformable and a rigid body*); and
- small sliding, which assumes that although two bodies may undergo large motions, there will be relatively little sliding of one surface along the other (see *Small-sliding interaction between bodies*).

You can choose between node-to-surface contact discretization and true surface-to-surface contact discretization for each of the above tracking approaches.

Formulations for General Contact

By default, general contact in Abaqus/Standard uses the finite-sliding, surface-to-surface contact formulation. This formulation can also be used for contact pairs, but it is not the default. The discussions in this section of finite-sliding, surface-to-surface contact apply equally to general contact and to contact pairs.

In a general contact domain the main and secondary roles are assigned to surfaces automatically, although it is possible to override these default assignments. The behavior of main surfaces and secondary surfaces is consistent across general contact and contact pair interactions. The specification of main and secondary surfaces in a general contact domain is covered in detail in *Contact Controls Specific to General Contact in Abaqus/Standard*.

Discretization of Contact Pair Surfaces

Abaqus/Standard applies conditional constraints at various locations on interacting surfaces to simulate contact conditions. The locations and conditions of these constraints depend on the contact discretization used in the overall contact formulation. Abaqus/Standard offers two contact discretization options: a traditional "node-to-surface" discretization and a true "surface-to-surface" discretization.

Node-to-Surface Contact Discretization

With traditional node-to-surface discretization the contact conditions are established such that each "secondary" node on one side of a contact interface effectively interacts with a point of projection on the "main" surface on the opposite side of the contact interface (see *Figure 1*). Thus, each contact condition involves a single secondary node and a group of nearby main nodes from which values are interpolated to the projection point.

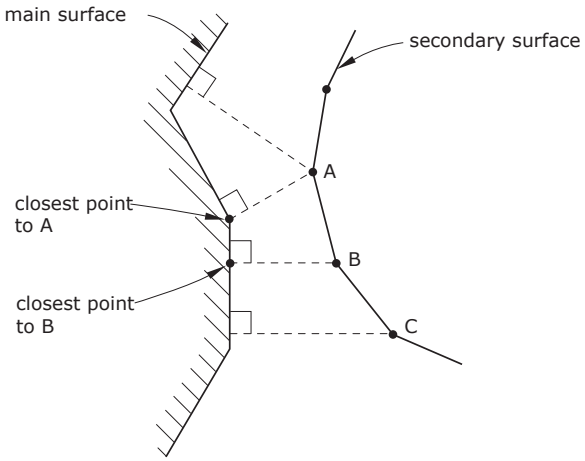


Figure 1: Node-to-surface contact discretization.

Traditional node-to-surface discretization has the following characteristics:

- The secondary nodes are constrained not to penetrate into the main surface; however, the nodes of the main surface can, in principle, penetrate into the secondary surface (for example, see the case on the upper-right of *Figure 2*).

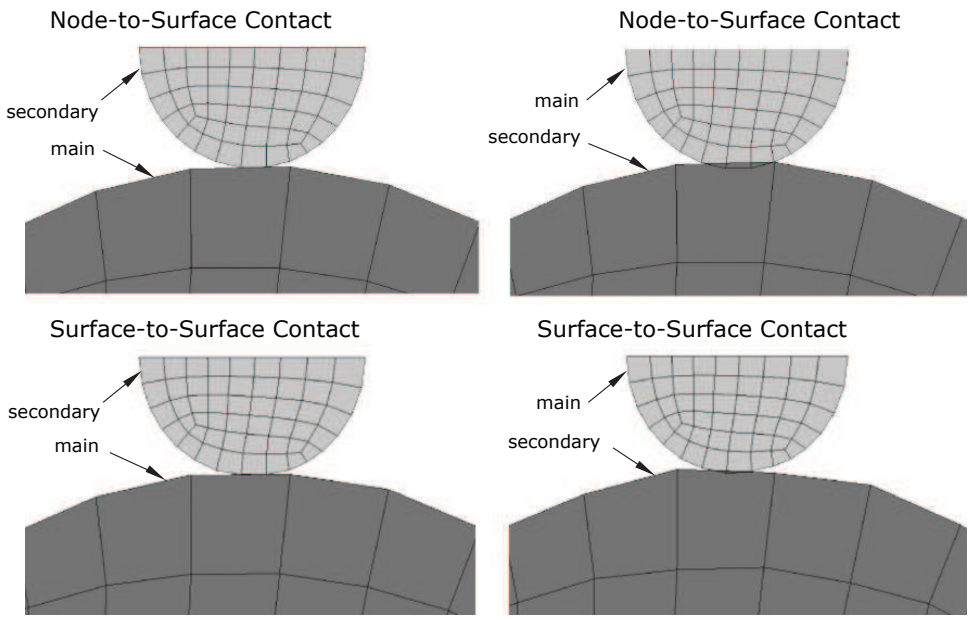


Figure 2: Comparison of contact enforcement for different main-secondary surface assignments with node-to-surface and surface-to-surface contact discretizations.

- The contact direction is based on the normal of the main surface.
- The only information needed for the secondary surface is the location and surface area associated with each node; the direction of the secondary surface normal and secondary surface curvature are not relevant. Thus, the secondary surface can be defined as a group of nodes—a node-based surface.
- Node-to-surface discretization is available even if a node-based surface is not used in a contact pair definition.

Surface-to-Surface Contact Discretization

Surface-to-surface discretization considers the shape of both the secondary and main surfaces in the region of contact constraints. Surface-to-surface discretization has the following key characteristics:

- The surface-to-surface formulation enforces contact conditions in an average sense over regions nearby secondary nodes rather than only at individual secondary nodes. The averaging regions are approximately centered on secondary nodes, so each contact constraint will predominantly consider one secondary node but will also consider adjacent secondary nodes. Some penetration may be observed at individual nodes; however, large, undetected penetrations of main nodes into the secondary surface do not occur with this discretization. *Figure 2* compares contact enforcement for node-to-surface and surface-to-surface contact for an example with dissimilar mesh refinement on the contacting bodies.
- The contact direction is based on an average normal of the secondary surface in the region surrounding a secondary node.
- Surface-to-surface discretization is not applicable if a node-based surface is used in the contact pair definition.

Choosing a Contact Discretization

In general, surface-to-surface discretization provides more accurate stress and pressure results than node-to-surface discretization if the surface geometry is reasonably well represented by the contact surfaces. *Figure 3* shows an example of improved contact pressure accuracy with surface-to-surface contact compared to node-to-surface contact.

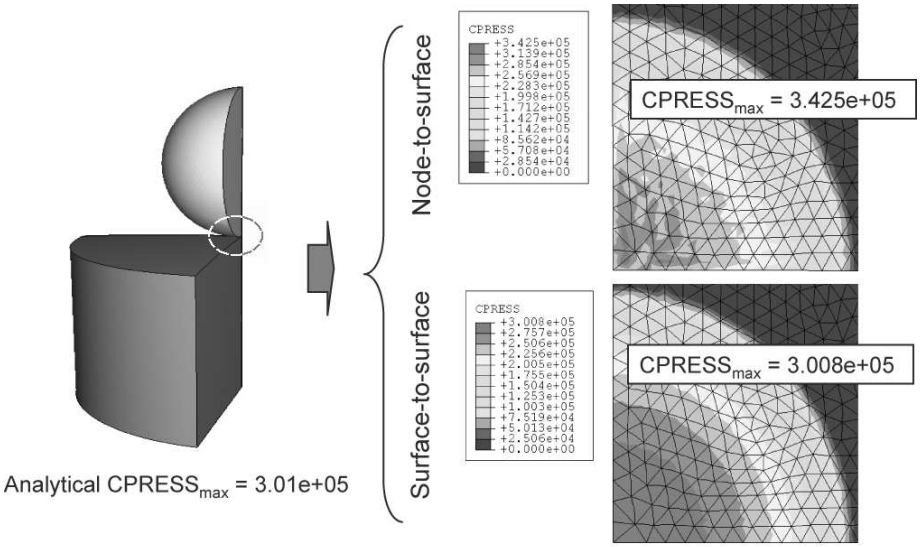


Figure 3: Comparison of contact pressure accuracy for node-to-surface and surface-to-surface contact discretizations.

Since node-to-surface discretization simply resists penetrations of secondary nodes into the main surface, forces tend to concentrate at these secondary nodes. This concentration leads to spikes and valleys in the distribution of pressure across the surface. Surface-to-surface discretization resists penetrations in an average sense over finite regions of the secondary surface, which has a smoothing effect. As the mesh is refined, the discrepancies between the discretizations lessen, but for a given mesh refinement the surface-to-surface approach tends to provide more accurate stresses.

Contact using surface-to-surface discretization is also less sensitive to main and secondary surface designations than node-to-surface contact (see *Choosing the Main and Secondary Surface Roles in a Two-Surface Contact Pair* below). *Figure 4* shows a simple model involving two blocks with dissimilar mesh densities.

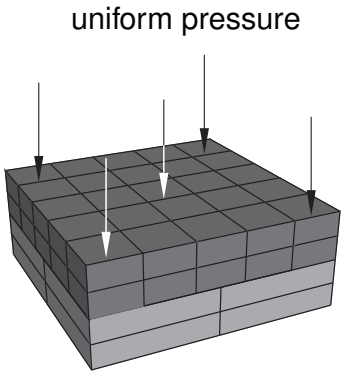


Figure 4: Test model for comparison of different main and secondary surface designations.

The bottom block is fixed to the ground, and a uniform pressure of 100 Pa is applied to the top face of the top block. Analytically, the top block should exert a uniform pressure of 100 Pa on the bottom block across the entire contact interface. *Table 1* compares the Abaqus analysis results for different contact discretizations and secondary surface designations.

Table 1: Error (from analytical results) for various discretization/secondary surface combinations.

Contact discretization	Secondary Surface	Maximum error in CPRESS
Node-to-surface	Top block	13%
	Bottom block	31%
Surface-to-surface	Top block	~1%
	Bottom block	~1%

If the surface geometry is not well-represented due to the use of a coarse mesh, significant inaccuracies can exist regardless of whether surface-to-surface contact or node-to-surface contact is used. In some cases surface smoothing techniques available for surface-to-surface contact can significantly improve solutions obtained with a coarse mesh. See *Smoothing Contact Surfaces in Abaqus/Standard* for a discussion of surface smoothing options for surface-to-surface contact.

Surface-to-surface discretization generally involves more nodes per constraint and can, therefore, increase solution cost. In most applications the extra cost is fairly small, but the cost can become significant in some cases. The following factors (especially in combination) can lead to surface-to-surface contact being costly:

- A large fraction of the model is involved in contact.
- The main surface is more refined than the secondary surface.
- Multiple layers of shells are involved in contact, such that the main surface of one contact pair acts as the secondary surface of another contact pair.

The surface-to-surface formulation is primarily intended for common situations in which normal directions of contacting surfaces are approximately opposite. The node-to-surface contact formulation is often preferable for treating contact involving feature edges or corners if the respective secondary and main facet normal directions are not approximately opposite in the active contact region.

Contact Tracking Approaches

In Abaqus/Standard there are two tracking approaches to account for the relative motion of two interacting surfaces in mechanical contact simulations.

The Finite-Sliding Tracking Approach

Finite-sliding contact is the most general tracking approach and allows for arbitrary relative separation, sliding, and rotation of the contacting surfaces. For finite-sliding contact the connectivity of the currently active contact constraints changes upon relative tangential motion of the contacting surfaces. For a detailed description of how Abaqus/Standard calculates finite-sliding contact, see *Using the Finite-Sliding Tracking Approach* later in this section.

The Small-Sliding Tracking Approach

Small-sliding contact assumes that there will be relatively little sliding of one surface along the other and is based on linearized approximations of the main surface per constraint. The groups of nodes involved with individual contact constraints are fixed throughout the analysis for small-sliding contact, although the active/inactive status of these constraints typically can change during the analysis. You should consider using small-sliding contact when the approximations are reasonable, due to computational savings and added robustness. For a detailed description of how Abaqus/Standard calculates small-sliding contact, see *Using the Small-Sliding Tracking Approach* later in this section.

Choosing the Main and Secondary Surface Roles in a Two-Surface Contact Pair

Abaqus/Standard enforces the following rules related to the assignment of the main and secondary roles for contact surfaces:

- Analytical rigid surfaces and rigid-element-based surfaces must always be the main surface.
- A node-based surface can act only as a secondary surface and always uses node-to-surface contact.
- Secondary surfaces must always be attached to deformable bodies or deformable bodies defined as rigid.
- Both surfaces in a contact pair cannot be rigid surfaces with the exception of deformable surfaces defined as rigid (see *Rigid Body Definition*).

When both surfaces in a contact pair are element-based and attached to either deformable bodies or deformable bodies defined as rigid, you have to choose which surface will be the main surface and which will be the secondary surface. This choice is particularly important for node-to-surface contact. Generally, if a smaller surface contacts a larger surface, it is best to choose the smaller surface as the secondary surface. If that distinction cannot be made, the main surface should be chosen as the surface of the stiffer body or as the surface with the coarser mesh if the two surfaces are on structures with comparable stiffnesses. The stiffness of the structure and not just the material should be considered when choosing the main and secondary surfaces. For example, a thin sheet of metal may be less stiff than a larger block of rubber even though the steel has a larger modulus than the rubber material. If the stiffness and mesh density are the same on both surfaces, the preferred choice is not always obvious.

The choice of main and secondary roles typically has much less effect on the results with a surface-to-surface contact formulation than with a node-to-surface contact formulation. However, the assignment of main and secondary roles can have a significant effect on performance with surface-to-surface contact if the two surfaces have dissimilar mesh refinement; the solution can become quite expensive if the secondary surface is much coarser than the main surface.

Fundamental Choices Affecting the Contact Formulation

Your choice of contact discretization and tracking approach have considerable impact on an analysis. In addition to the qualities already discussed, certain combinations of discretizations and tracking approaches have their own characteristics and limitations associated with them. These characteristics are summarized in *Table 2*. You should also consider the solution costs associated with the various contact formulations.

Table 2: Comparison of contact formulation characteristics.

Characteristic	Contact formulation			
	Node-to-surface		Surface-to-surface	
	Finite-sliding	Small-sliding	Finite-sliding	Small-sliding
Account for shell thickness by default	No	Yes	Yes	Yes
Allow self-contact	Yes	No	Yes	No
Allow double-sided surfaces	Secondary surface only	Secondary surface only	Yes[1]	Yes
Surface smoothing by default	Some smoothing of main surface	Yes for anchor points; each constraint uses flat approximation of main surface	No	No for anchor points; each constraint uses flat approximation of main surface
Default constraint enforcement method	Augmented Lagrange method for 3D	Direct method	Penalty method	Direct method

Characteristic	Contact formulation			
	Node-to-surface		Surface-to-surface	
	Finite-sliding	Small-sliding	Finite-sliding	Small-sliding
	self-contact; otherwise, direct method			
Ensure moment equilibrium for offset reference surfaces with friction	No	No	Yes	Yes

[1]Double-sided main surfaces are allowed with the finite-sliding, surface-to-surface formulation only if the path-based tracking algorithm is used (see *Path-Based Versus State-Based Tracking Algorithms*). Double-sided secondary surfaces are allowed with both tracking algorithms if the main surface is not user defined.

Accounting for Shell Thickness

Most contact formulations will account for the surface thickness of a shell when calculating contact constraints. However, the finite-sliding, node-to-surface formulation will not account for shell thicknesses. These calculations are discussed in more detail in *Accounting for Shell and Membrane Thickness*.

Allowing for Self-Contact

Self-contact is typically the result of large deformation in a model. It is often difficult to predict which regions will be involved in the contact or how they will move relative to each other. Therefore, self-contact cannot use the small-sliding tracking approach.

Allowing Double-Sided Surfaces

Doubled-sided contact surfaces based on shell-like elements are allowed to act as secondary and/or main surfaces for the surface-to-surface contact formulation by default and are allowed to act as the secondary surface for the node-to-surface contact formulation. For a shell-like surface to act as the main surface for the surface-to-surface formulation with the optional state-based tracking algorithm (see *Path-Based Versus State-Based Tracking Algorithms* below) or for the node-to-surface contact formulation, the surface must be defined as single-sided (see *Defining Single-Sided Surfaces* and *Orientation Considerations for Shell-Like Surfaces* for more information).

Surface Smoothing

When using node-to-surface discretization, corners or small protrusions of a jagged main surface are allowed to penetrate the spaces between nodes in the node-based surface. It is sometimes possible for a secondary node sliding along the main surface to snag on these corners. Therefore, Abaqus/Standard automatically smooths the main surface for contact calculations utilizing node-to-surface discretization to minimize this phenomenon. The details are discussed further in *Smoothing Main Surfaces for the Finite-Sliding, Node-to-Surface Formulation* later in this section.

No surface smoothing occurs by default when using surface-to-surface discretization. Surface-to-surface discretization considers contact conditions in an average sense over a finite region, which tends to alleviate problems associated with small protrusions of the main surface penetrating the secondary surface and introduces some inherent smoothing characteristics at the constraint level. However, this inherent smoothing typically does not significantly mitigate errors associated with poor geometric representations of curved surfaces when a relatively coarse mesh is used. In some cases nondefault circumferential or spherical surface smoothing methods available for surface-to-surface contact can significantly improve solutions obtained with a coarse mesh (see *Smoothing Contact Surfaces in Abaqus/Standard*).

Constraint Enforcement Methods

In many cases Abaqus/Standard strictly enforces the contact constraints discussed previously by default. However, strict enforcement of contact constraints can sometimes lead to overconstraint issues (for example, see *Overconstraint Checks*) or convergence difficulty. To address these issues and allow for decreased solution cost with typically minimal sacrifice to solution accuracy, Abaqus/Standard also provides penalty-based constraint enforcement methods. The numerical constraint enforcement methods (and defaults) are discussed in detail in *Contact Constraint Enforcement Methods in Abaqus/Standard*.

Moment Equilibrium

Based on Newton's third law of motion, contact forces should be self-equilibrating; that is, the net contact forces acting on the respective surfaces for each active contact constraint should be equal and opposite and effectively act through a common point. Contact constraints based on surface-to-surface contact discretization always exhibit this characteristic. Contact constraints based on node-to-surface discretization always generate zero net force, but under certain circumstances can generate a net moment in the numerical solution. Frictional forces associated with node-to-surface contact constraints will generate net moment if an offset exists between the respective reference surfaces. The following factors can contribute to a normal-direction offset between nodes of respective contact surfaces while contact constraints are active:

- The presence of a softened pressure-versus-overclosure behavior (due to a user-specified, softened pressure-overclosure model or use of a constraint enforcement method, such as the penalty method, that exhibits numerical softening.
- Contact calculations accounting for shell or membrane thicknesses (which is not allowed with the finite-sliding, node-to-surface formulation).
- User-specified initial contact clearances (see *Additional Contact Initialization Options for Small-Sliding Contact in Abaqus/Standard*).
- Various usages of special-purpose contact elements, such as tube-to-tube contact elements (see *Contact Modeling with Elements* and *Tube-to-Tube Contact Elements*), result in some normal distance between nodes that interact with each other.

While undesirable, the net moment that sometimes occurs with node-to-surface contact constraints is typically not significantly detrimental to the analysis results.

Effect of the Contact Discretization Method on Solution Cost

There is no easy way to predict which contact discretization method will result in lower overall solution cost. Basic trends include:

- Node-to-surface contact discretization tends to be less costly per iteration than surface-to-surface contact discretization (because surface-to-surface contact discretization generally involves more nodes per constraint).
- Contact conditions with finite-sliding contact tend to converge in fewer iterations with surface-to-surface contact discretization than with node-to-surface contact discretization (because surface-to-surface contact discretization has more continuous behavior upon sliding).

Using the Finite-Sliding Tracking Approach

The finite-sliding tracking approach allows for arbitrary separation, sliding, and rotation of the surfaces. Abaqus/Standard contact pairs use a finite-sliding, node-to-surface contact formulation by default. General contact in Abaqus/Standard always uses a finite-sliding, surface-to-surface contact formulation.

Example

Consider the case shown in *Figure 5*, with surface ASURF acting as the secondary surface to surface BSURF in a finite-sliding, node-to-surface contact pair.

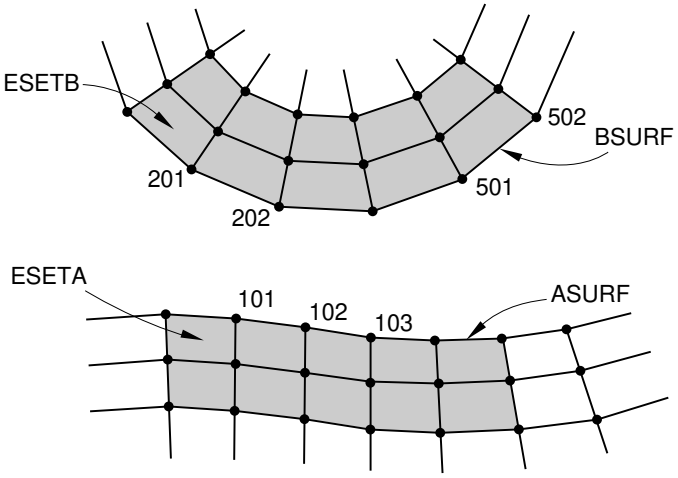


Figure 5: Contacting bodies.

In this example secondary node 101 may come into contact anywhere along the main surface BSURF. While in contact, it is constrained to slide along BSURF, irrespective of the orientation and deformation of this surface. This behavior is possible because Abaqus/Standard tracks the position of node 101 relative to the main surface BSURF as the bodies deform. *Figure 6* shows the possible evolution of the contact between node 101 and its main surface BSURF.

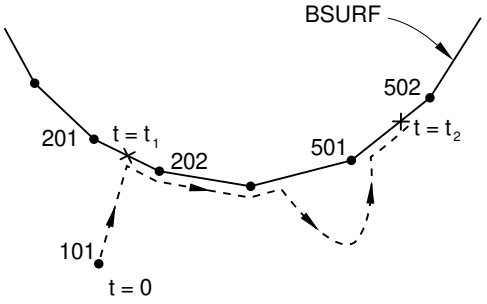


Figure 6: Trajectory of node 101 in finite-sliding contact.

Node 101 is in contact with the element face with end nodes 201 and 202 at time t_1. The load transfer at this time occurs between node 101 and nodes 201 and 202 only. Later on, at time t_2, node 101 may find itself in contact with the element face with end nodes 501 and 502. Then the load transfer will occur between node 101 and nodes 501 and 502.

Path-Based Versus State-Based Tracking Algorithms

Brief descriptions of the tracking algorithms available in Abaqus/Standard are provided below so that you can be aware of their characteristics and available options.

Path-Based Tracking Algorithm

The “path-based” tracking algorithm carefully considers the relative paths of points on the secondary surface with respect to the main surface within each increment and allows for double-sided shell and membrane main

surfaces. The path-based tracking algorithm is available only for finite-sliding, surface-to-surface contact interactions involving element-based main surfaces and is the default for those interactions. The path-based algorithm is sometimes more effective than the state-based algorithm for analyses involving self-contact or large incremental relative motion.

Input File Usage: Use the following option to specify use of the path-based tracking algorithm:

**CONTACT PAIR*, INTERACTION=*interaction_property_name*, TYPE=SURFACE TO SURFACE, TRACKING=PATH

Abaqus/CAE Usage: Interaction module: surface-to-surface contact or self-contact interaction editor: **Discretization method**: **Surface to surface**, **Contact tracking**: **Two configurations (path)**

State-Based Tracking Algorithm

The "state-based" tracking algorithm updates the tracking state based on the tracking state associated with the beginning of the increment together with geometric information associated with the predicted configuration. This algorithm is well-suited for most finite-sliding analyses but requires the use of single-sided surfaces and occasionally has difficulty tracking large incremental motion. State-based tracking may miss detecting contact if the incremental relative motion exceeds the dimensions of the main surface or if the incremental motion cuts across corners of the main surface; specifying an upper bound for the increment size helps avoid these problems. The state-based tracking algorithm is:

- the only tracking algorithm available for finite-sliding, node-to-surface contact pairs;
- the only tracking algorithm available for finite-sliding contact interactions involving an analytical rigid main surface;
- a non-default option for finite-sliding, surface-to-surface contact pairs involving an element-based main surface.

Input File Usage: Use the following option to specify use of the state-based tracking algorithm:

**CONTACT PAIR*, INTERACTION=*interaction_property_name*, TYPE=SURFACE TO SURFACE, TRACKING=STATE

Abaqus/CAE Usage: Interaction module: surface-to-surface contact or self-contact interaction editor: **Discretization method**: **Surface to surface**, **Contact tracking**: **Single configuration (state)**

Smoothing Main Surfaces for the Finite-Sliding, Node-to-Surface Formulation

The finite-sliding, node-to-surface contact formulation requires that main surfaces have continuous surface normals at all points. Convergence problems can result if main surfaces that do not have continuous surface normals are used in finite-sliding, node-to-surface contact analyses; secondary nodes tend to get "stuck" at points where the main surface normals are discontinuous. Abaqus/Standard automatically smooths the surface normals of element-based main surfaces (see *Smoothing Deformable Main Surfaces and Rigid Surfaces Defined with Rigid Elements* below) used in finite-sliding, node-to-surface contact simulations, including those modeled with slide lines. You are expected to create smooth analytical rigid surfaces (see *Analytical Rigid Surface Definition*). No such smoothing of main surface normals is needed with the finite-sliding, surface-to-surface formulation.

Smoothing Deformable Main Surfaces and Rigid Surfaces Defined with Rigid Elements

For finite-sliding, node-to-surface contact simulations with planar or axisymmetric deformable main surfaces, Abaqus/Standard will smooth any discontinuous transitions between two first-order element faces with parabolic curves. Discontinuous transitions between two second-order element faces are smoothed with cubic curves connecting two points located on the element's faces. This smoothing is shown in *Figure 7* for first-order elements (linear segments) and in *Figure 8* for second-order elements (parabolic segments). For finite-sliding, node-to-surface simulations with three-dimensional deformable main surfaces and rigid main surfaces using rigid elements, Abaqus/Standard will smooth any discontinuous surface normal transitions between the main surface facets.

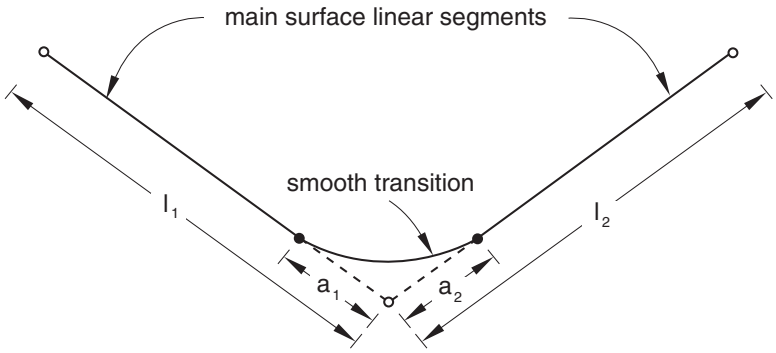


Figure 7: Smoothing between linear segments.

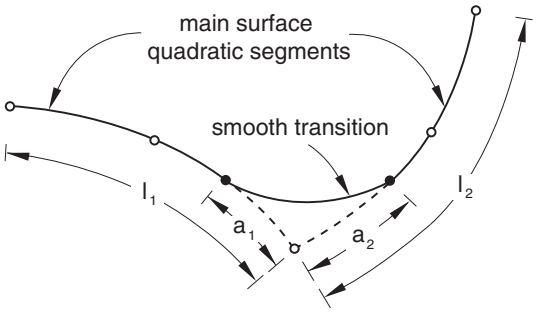


Figure 8: Smoothing between quadratic segments.

You can control the degree of smoothing of the main surface in node-to-surface contact simulations or in analyses using slide lines and contact elements by specifying a fraction f. The default value of f is 0.2.

For planar or axisymmetric deformable main surfaces, $f = a_1/\ell_1 = a_2/\ell_2$, where ℓ_1 and ℓ_2 are the lengths of the element facets that join at the surface node and $f < 0.5$ (see *Figure 7* and *Figure 8*). Abaqus/Standard will construct either a parabolic or a cubic segment between two points at distances a_1 and a_2 from the node at which the discontinuity exists; this smoothed segment will be used in the contact calculations. Thus, the contact surface will differ from the faceted element geometry. Smoothing affects only segments where the normal to the deformable main surface is discontinuous at the node joining two elements: it does not affect the two segments adjacent to the midside nodes on second-order element faces.

For three-dimensional, element-based main surfaces, f is defined as a fraction of the dimension of a facet as shown in *Figure 9*. The normal vector of a point within the region bounded by the dashed lines is computed to be normal to the facet. Outside this region the normal is smoothed with respect to the adjacent facets, using a generalization of the two-dimensional approach shown in *Figure 7* and *Figure 8*. The physical geometry of a three-dimensional facet is not smoothed; only the surface normal definitions associated with the facet are affected by the smoothing operation. The implementation of the normal-direction smoothing algorithm is slightly different for surfaces based on rigid type elements (see *Rigid Elements*) than other element types. This difference typically

has minimal effect on the convergence behavior or solution results; however, for example, different solution behavior may occasionally be observed between otherwise identical analyses in which a rigid body is modeled with R3D4 elements in one case and S4R elements assigned to a rigid body in another case.

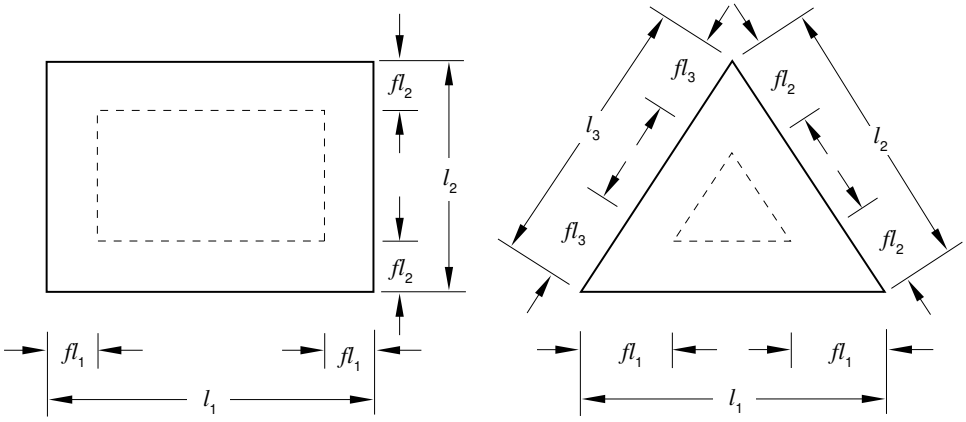


Figure 9: Smoothing of a three-dimensional main surface.

Input File Usage: Use the following option for node-to-surface contact simulations:

**CONTACT PAIR*, INTERACTION=*interaction_property_name*, SMOOTH=*f*

Use the following option when using slide lines and contact elements:

**SLIDE LINE*, ELSET=*name*, SMOOTH=*f*

Abaqus/CAE Usage: Interaction module: **Interaction->Create**: **Surface-to-surface contact (Standard)** or **Self-contact (Standard)**: **Degree of smoothing for main surface:** *f*

Smoothing a Deformable Main Surface along Symmetry Edges

When a two-dimensional or axisymmetric deformable main surface ends at a symmetry plane and node-to-surface discretization is used, Abaqus/Standard will smooth and calculate the proper surface normals and tangent planes of the end segment if the boundary condition at the symmetry end is specified with the symmetry "type" boundary XSYMM or YSYMM. This smoothing procedure is accomplished by reflecting the end segment about the symmetry plane and constructing either a parabolic or a cubic segment between the end segment and the reflected segment. Thus, the contact surface may differ from the faceted element geometry near the end. Abaqus/Standard will automatically adjust the surface normal and tangent planes at $r = 0$ of an axisymmetric main surface regardless of whether a symmetry boundary condition is defined. The finite-sliding, surface-to-surface formulation has no special treatment for surfaces ending at a symmetry plane. See *Modifying the Main Surface Normals* for a discussion of how the small-sliding, node-to-surface formulation treats main surfaces ending at a symmetry plane. See *Small-Sliding, Surface-to-Surface Contact* for a discussion of how the small-sliding, node-to-surface formulation treats secondary surfaces ending at a symmetry plane.

Overriding the Default Smoothing Behavior for Finite-Sliding, Node-to-Surface Contact

To model a main surface with corners in two dimensions (fold lines in three dimensions), break the surface into multiple surfaces. This technique prevents Abaqus/Standard from smoothing out the corners or fold lines and allows Abaqus/Standard to introduce constraints associated with each surface if a secondary node is in contact with an interior corner or fold in the main surface.

To accurately model the main surface with a corner shown in *Figure 10*, you must define two contact pairs: the first contact pair has `ASURF` as the secondary surface and `BSURFA` as the main surface; the second contact pair has `ASURF` as the secondary surface and `BSURFB` as the main surface.

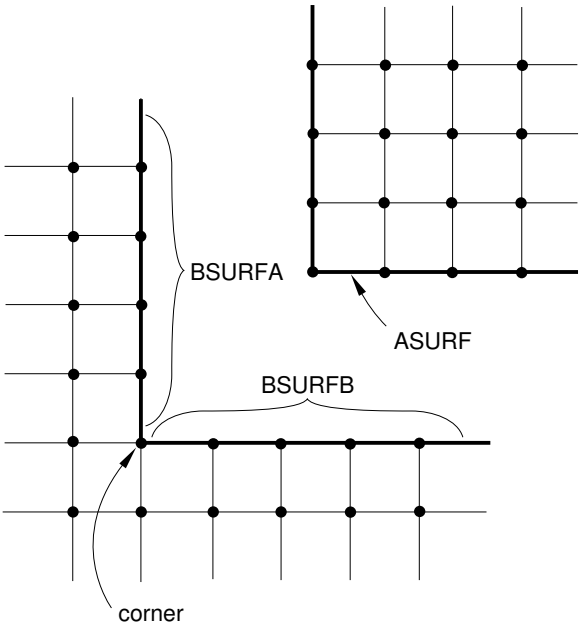


Figure 10: main surface with a corner.

Finite Sliding in a Geometrically Linear Analysis

Finite-sliding simulations usually include nonlinear geometric effects because such simulations generally involve large deformations and large rotations. However, it is also possible to use the finite-sliding tracking approach in a geometrically linear analysis (see *Geometric Nonlinearity*). The load transfer paths between the surfaces and the contact direction are updated in finite-sliding, geometrically linear analyses. This capability is useful for analyzing finite sliding between two stiff bodies that do not undergo large rotations.

Unsymmetric Terms in Finite-Sliding Contact Simulations

Normal contact constraints due to node-to-surface discretization produce unsymmetric terms in the system of equations when three-dimensional faceted surfaces come in contact. These terms have a strong effect on the convergence rate in regions on the main surfaces with large differences in surface normals between facets.

Normal contact constraints due to surface-to-surface discretization produce unsymmetric terms in both two- and three-dimensional cases. These terms have a strong effect on the convergence rate in regions where the main and secondary surfaces are not parallel to each other.

In both cases you should use the unsymmetric solution scheme for the step to improve the convergence rate of the simulation (see *Matrix Storage and Solution Scheme in Abaqus/Standard*).

Contact simulations that involve strong frictional effects can also produce unsymmetric terms. See *Unsymmetric Terms in the System of Equations* for details.

Using the Small-Sliding Tracking Approach

For a large class of contact problems the general tracking of the finite-sliding approach is unnecessary, even though geometric nonlinearity may need to be considered. Abaqus/Standard provides a small-sliding tracking approach for such problems. For geometrically nonlinear analyses this formulation assumes that the surfaces

may undergo arbitrarily large rotations but that a secondary node will interact with the same local area of the main surface throughout the analysis. For geometrically linear analyses the small-sliding approach reduces to an infinitesimal-sliding and rotation approach, in which it is assumed that both the relative motion of the surfaces and the absolute motion of the contacting bodies are small.

Abaqus/Standard attempts to associate a planar approximation of the main surface with each secondary node of a small-sliding contact pair. Contact interactions are considered between a given secondary node (or region nearby a given secondary node for the surface-to-surface formulation) and the associated local tangent plane. An example for the small-sliding, node-to-surface formulation is shown in *Figure 11* (for example, the secondary node is typically constrained not to penetrate this local tangent plane). Each local tangent plane, which is a line in two dimensions, is defined by an anchor point, $\mathbf{X_0}$, on the main surface and an orientation vector at the anchor point (see *Figure 11*).

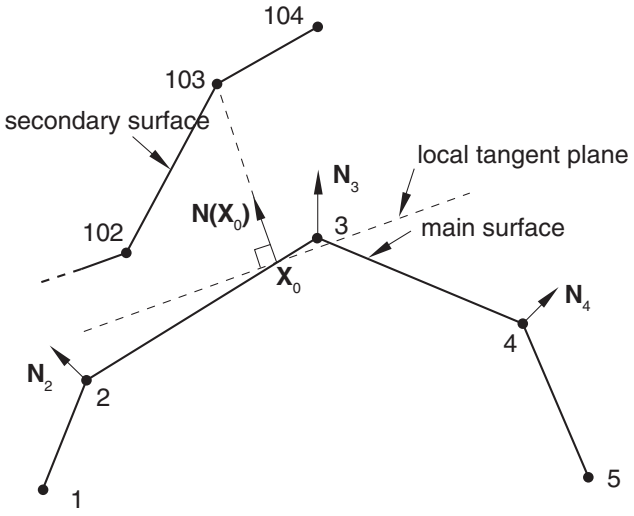


Figure 11: Definition of the anchor point and local tangent plane used by the small-sliding, node-to-surface formulation for node 103.

The algorithm used to define anchor points is described below. If an anchor point cannot be determined for a particular secondary node, no contact constraint will be enforced for that secondary node.

Having a local tangent plane for each secondary node means that for the small-sliding tracking approach Abaqus/Standard does not have to monitor secondary nodes for possible contact along the entire main surface. Therefore, small-sliding contact is generally less expensive computationally than finite-sliding contact. The cost savings are often most dramatic in three-dimensional contact problems.

Small-Sliding, Node-to-Surface Contact

For node-to-surface contact Abaqus/Standard chooses the anchor point of a secondary node's local tangent plane such that the vector from the anchor point to the secondary node coincides with a smoothly varying normal vector on the main surface. The anchor point is chosen before the analysis starts using the initial configuration of the model.

Smoothly Varying Main Surface Normals

The algorithm requires that the main surface have a smoothly varying normal vector $\mathbf{N}(\mathbf{x})$, where $\mathbf{x}$ is any point on the main surface. The first step in defining $\mathbf{N}(\mathbf{x})$ is to construct the unit normal vectors at each node of the main surface. Abaqus/Standard forms these nodal normals by averaging the normals of the element faces making

up the main surface; only the element faces in the surface definition will contribute to the nodal normals and, thus, to $\mathbf{N}(\mathbf{x})$. Abaqus/Standard uses the initial nodal coordinates to compute these normals.

Figure 11 shows the nodal unit normals for a main surface, the anchor point $\mathbf{X}_0$, and the local tangent plane associated with secondary node 103. Abaqus/Standard uses the nodal unit normals $\mathbf{N}_2$ and $\mathbf{N}_3$, along with the shape functions of the element containing the two nodes, to construct $\mathbf{N}(\mathbf{x})$ on the 2–3 element face. Abaqus/Standard chooses the anchor point $\mathbf{X}_0$ of the local tangent plane for node 103 so that $\mathbf{N}(\mathbf{X}_0)$ passes through node 103. $\mathbf{N}(\mathbf{X}_0)$ is the contact direction for secondary node 103 and defines the orientation of the local tangent plane. In this example, as in many cases, the local tangent plane is only an approximation of the actual mesh geometry.

Modifying the Main Surface Normals

Defining user-specified nodal normals on the main surface (see *Normal Definitions at Nodes*) will improve the local tangent planes calculated for the small-sliding, node-to-surface formulation in some cases. For example, a default nodal normal corresponding to an average normal among adjacent facets can cause significant deviation from the true surface normal direction at perimeter nodes, as shown in *Figure 12*. The nodal normal $\mathbf{N}_1$ does not point along the symmetry plane, which means that secondary node 100 will never intersect the main surface. In a small-sliding problem if a secondary node fails to intersect the main surface at the start of the analysis, it will be free to penetrate the main surface because no local tangent plane will be formed.

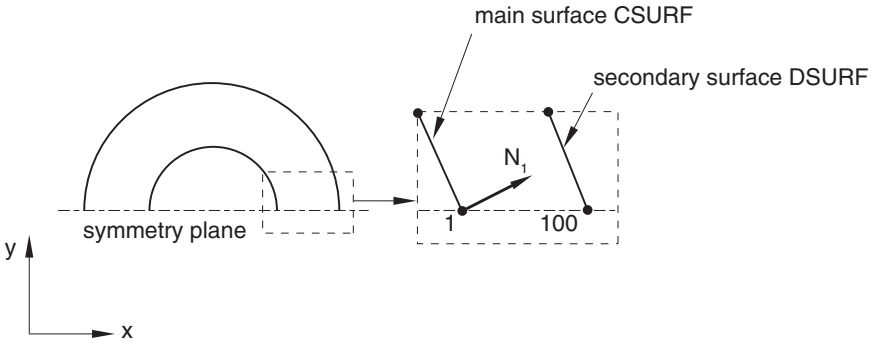


Figure 12: Main surface normal at node 1 in a small-sliding model of concentric cylinders. With the default $\mathbf{N}_1$ secondary node 100 will never contact `CSURF`.

Defining a user-specified normal (1.00E+00, 0.00E+00, 0.00E+00) at node 1 on the main surface `CSURF` will correct the problem, as shown in *Figure 13*. This method allows secondary node 100 to see the main surface, and the correct contact normal direction will be used. Main surface normals at perimeter nodes are adjusted automatically to lie along the symmetry plane if boundary conditions are specified at these nodes in symmetry "type" format (XSYMM, YSYMM, or ZSYMM—see *Boundary Conditions*).

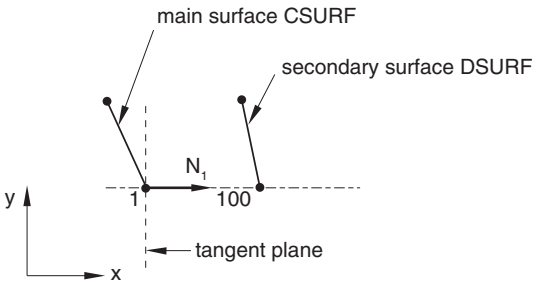


Figure 13: The modified main surface normal at node 1 of `CSURF` now allows secondary node 100 to contact `CSURF`.

Small-Sliding, Surface-to-Surface Contact

A key difference with the surface-to-surface approach is that more than one secondary node is involved in each contact constraint (except when the secondary surface is based on gasket elements, as discussed below). This is related to the fact that the surface-to-surface formulation enforces contact conditions in an average sense over regions nearby secondary nodes rather than only at individual secondary nodes (see *Surface-to-Surface Contact Discretization* above). The small-sliding, surface-to-surface contact formulation is a limit case of the finite-sliding, surface-to-surface formulation, using a planar approximation of the main surface per averaging region of the secondary surface. The constraint participation factors for the secondary nodes remain constant for small-sliding contact. The effective center-of-action on the secondary surface per contact constraint may differ slightly from the location of the predominant secondary node associated with the constraint.

A special version of the small-sliding, surface-to-surface formulation is used if the secondary surface is based on gasket elements to avoid a tendency to trigger unstable deformation modes in the gasket elements. This special formulation has only one secondary node per contact constraint and preserves the accuracy advantages of the surface-to-surface formulation, but it is not well-suited for extension to finite-sliding and is otherwise not as generally applicable as the regular small-sliding, surface-to-surface formulation. (The finite-sliding, surface-to-surface formulation always uses multiple secondary nodes per constraint and is not recommended for contact involving gasket elements.)

The small-sliding, surface-to-surface contact formulation determines main anchor points and normal directions in a manner similar to that used by the small-sliding, node-to-surface contact formulation; however, there are some differences. For the surface-to-surface approach the anchor point approximately corresponds to the center of the zone on the main surface where the averaging region of the secondary projects onto the main surface. This projection occurs along the secondary surface normal direction. This method does not make use of smoothed main surface nodal normals. The anchor point location typically does not depend significantly on whether node-to-surface or surface-to-surface discretization is used, unless the surfaces are significantly separated and non-parallel in the initial configuration (in which case small-sliding contact may not be appropriate).

Abaqus/Standard automatically reverts to the node-to-surface approach for individual small-sliding contact constraints in the following circumstances, even if you have specified use of the surface-to-surface approach:

- if the secondary surface is a node-based surface;
- if the projection along the secondary surface normal direction does not intersect the main surface (but an anchor point can be found using the interpolated main surface normal direction algorithm discussed above for the small-sliding, node-to-surface formulation); or
- if single-sided secondary and main surfaces have surface normals in approximately the same direction.

For constraints based on surface-to-surface discretization it is not necessary that the constraint associated with a node on a symmetry plane is parallel to the symmetry plane. Hence, there is usually no need to specify specific normal directions. As in the case of node-to-surface contact, the contact direction points from the anchor point to the secondary node, and the tangent plane is normal to this direction. The contact normal for the small-sliding, surface-to-surface formulation is adjusted automatically to lie along the symmetry plane for each secondary node that has a boundary condition specified in symmetry “type” format (XSYMM, YSYMM, or ZSYMM—see *Boundary Conditions*).

Orientation of Local Tangent Planes

The local tangent plane is by definition orthogonal to the contact direction. You can override the default contact direction to specify a direction with a spatially varying clearance or overclosure definition (see *Specifying the Surface Normal for the Contact Calculations*).

Once the contact direction is defined, the orientation of the local tangent plane with respect to the main surface facet remains fixed. Because small-sliding contact considers nonlinear geometric effects, Abaqus/Standard continuously updates the orientation of the local tangent plane to account for the rotation and, assuming that the

main surface is deformable, the deformation of the main surface. The position of the anchor point relative to the surrounding nodes on the main surface facet does not change as the main surface deforms.

Load Transfer

In a small-sliding analysis each constraint can transfer load only to a limited number of nodes on the main surface. These nodes on the main surface are chosen based on their initial proximity to the anchor point. The magnitude of load transferred to each main surface node is based on proximity in the current, deformed configuration to the center-of-action on the secondary surface (which corresponds to a secondary node for the node-to-surface formulation). For example, in *Figure 11* node 103 transmits load to both nodes 2 and 3 on the main surface if node-to-surface discretization is used (if surface-to-surface discretization is used, load may be transmitted to additional nearby main nodes). Thus, if node 103 contacts the local tangent plane, a larger share of the force would be transmitted to the main surface node, 2 or 3, closer to the secondary node.

When the anchor point $\mathbf{X}_0$ corresponds to a node on the main surface, as is the case with secondary node 104 and main surface node 3 in *Figure 11*, the transmitted load for node-to-surface contact is shared by the node at $\mathbf{X}_0$ and all of the main surface nodes that share an adjacent surface facet with that node (additional main nodes may take part in the load transfer for surface-to-surface contact). In *Figure 11* the three main surface nodes sharing the force transmitted by secondary node 104 are nodes 2, 3, and 4.

As the center-of-action on the secondary surface for a constraint slides along its local tangent plane, Abaqus/Standard updates the distribution among the main surface nodes. However, no additional main surface nodes are ever added to the original list of nodes associated with a given small-sliding constraint. The constraint will continue to transmit load to the original list of main surface nodes, regardless of the sliding distance. *Figure 14* shows the potential problem that arises if small sliding is used but the relative tangential motion of the surfaces is not "small." It shows the possible evolution of contact between secondary node 101 in *Figure 5* and its main surface `BSURF`. Using the unit normal vectors $\mathbf{N}_{201}$ and $\mathbf{N}_{202}$, the anchor point $\mathbf{X}_0$ is found for secondary node 101; for the purposes of this example, assume that it lies at the midpoint of the 201–202 face. With this location of $\mathbf{X}_0$ the local tangent plane for node 101 is parallel with the 201–202 face. The load transfer always occurs between node 101 and nodes 201 and 202, no matter how far node 101 slides along the local tangent plane. Therefore, if node 101 moves as shown in *Figure 14*, it will continue to transmit load to nodes 201 and 202 when, in fact, it really slid off the mesh forming the main surface `BSURF`.

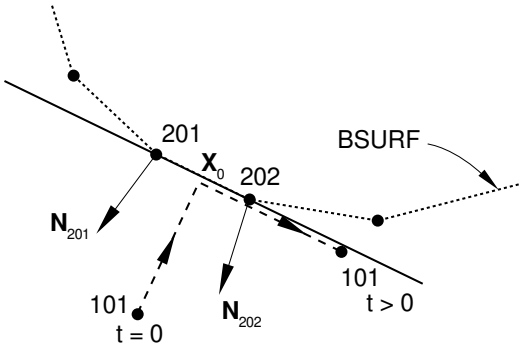


Figure 14: Excessive sliding in a small-sliding contact analysis.

What Can Be Considered Small Sliding

A contact pair in a small-sliding contact simulation should not grossly violate any of the assumptions or limitations outlined above. Adhere to the following guidelines:

- Secondary nodes should slide less than an element length from their corresponding anchor point and still be contacting their local tangent plane. If the main surface is highly curved, the secondary nodes should slide

only a fraction of an element length. The accumulated slip at a secondary node (CSLIP) can provide a good estimate of how far a secondary node has moved.

- The local tangent planes formed by Abaqus/Standard should be a good approximation of the mesh geometry; if necessary, define a user-specified normal (*Normal Definitions at Nodes*) to improve the smoothly varying main surface normal, $\mathbf{N}(\mathbf{x})$.
- The rotation and deformation of the main surface should not cause the local tangent planes to become a poor representation of the main surface during the course of the analysis.

Choosing the Main and Secondary Surfaces in Small-Sliding Problems

The basic guidelines given in *About Contact Pairs in Abaqus/Standard* should still be followed in a small-sliding simulation—the secondary surface should be the more refined surface or the surface on the more deformable body. However, in a small-sliding simulation more thought must be given when defining the main surface. With small-sliding contact each secondary node views the main surface as a flat surface, which can be significantly different than the true shape of the surface, even in the local region near the anchor point. In some cases the local tangent planes provide a good local approximation to the main surface in the initial configuration, but deformation and rotation of the main surface can reorient the local tangent planes such that they become a poor representation of the main surface. *Figure 15* shows an example where distortion of the main surface results in such a situation.

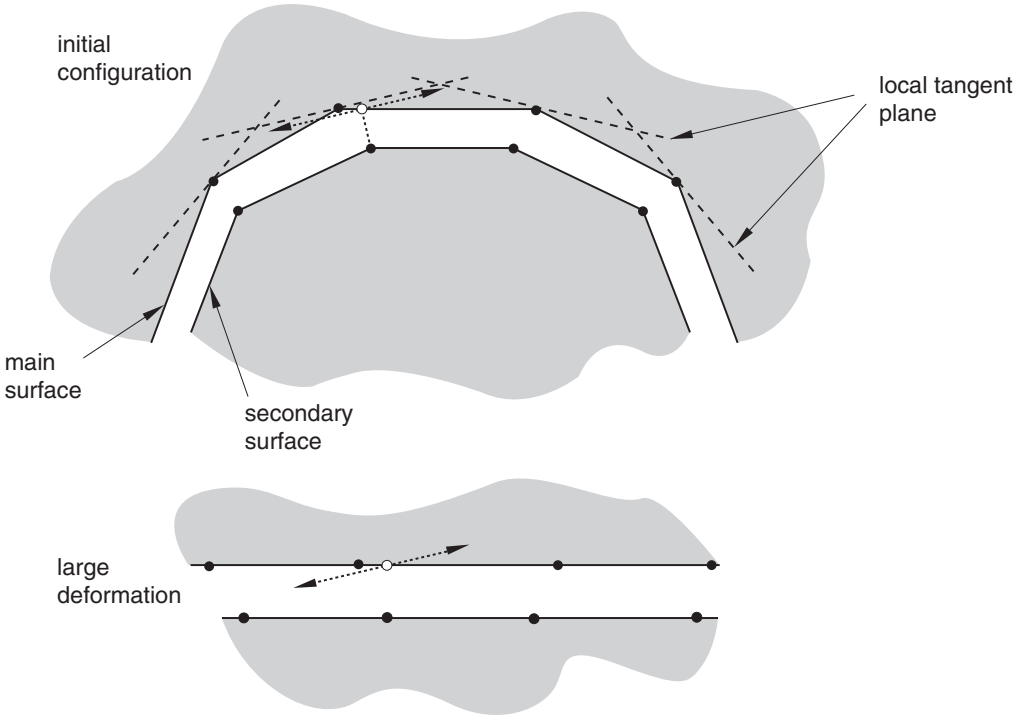


Figure 15: Main surface deformation in a small-sliding contact analysis can cause problems with the local tangent planes.

This problem can be minimized to some extent by using a more refined mesh on the main surface, thus providing more element faces to control the motion of the tangent planes. Excessive mesh refinement should not be necessary since only small sliding should occur.

Infinitesimal Sliding

As was mentioned before, the small-sliding tracking approach reduces to an infinitesimal-sliding tracking approach for geometrically linear analyses. Infinitesimal sliding assumes that both the relative motions of the

surfaces and the absolute motions of the model remain small. The orientations of the local tangent planes are not updated, and the load transfer paths and the weightings assigned to each main surface node remain constant during an infinitesimal-sliding simulation.

As in the case of small sliding, you can choose between node-to-surface and surface-to-surface discretizations with the infinitesimal-sliding tracking approach. The same user interface applies, and the default is node-to-surface discretization.

Local Tangent Directions on a Surface

Local tangent directions on a contact surface are a reference orientation by which Abaqus calculates tangential behavior in a contact interaction. Abaqus/Standard calculates the initial orientation of the two local tangent directions by default. The local tangent directions rotate with the contact surface in a geometrically nonlinear analysis.

Calculating the Initial Local Tangent Directions for a Two-Dimensional Surface

Two-dimensional and standard axisymmetric models have only one local tangent direction, $\mathbf{t_1}$. Abaqus/Standard defines the orientation of this direction by the cross product of the vector into the plane of the model (0., 0., 1.0) and the contact normal vector.

Models consisting of generalized axisymmetric bodies have a second local tangent direction, $\mathbf{t_2}$, to account for the component of slip associated with relative differences in circumferential twist between contacting bodies. The first local tangent direction at any point on the surface is always tangent to the main surface in the local *r–z* plane. The second local tangent direction is orthogonal to this plane in the local circumferential direction. For more information about generalized axisymmetric models, see *Generalized Axisymmetric Stress/Displacement Elements with Twist*.

Calculating the Initial Local Tangent Directions for a Three-Dimensional Surface

By default, Abaqus/Standard determines the initial orientation of the two local tangent directions, $\mathbf{t_1}$ and $\mathbf{t_2}$, using the following conventions:

Finite-sliding, surface-to-surface formulation

The default initial orientations of the two local tangent directions are based on the secondary surface normal, using the standard convention for calculating surface tangents (see *Conventions*) with the assumption that the contact normal corresponds to the negative normal to the secondary surface.

Finite-sliding, node-to-surface formulation

For contact involving a secondary surface based on three-dimensional beam-type elements, the first and second local tangent directions are defined along the length of the beam and transverse to the beam, respectively. For contact involving an analytical rigid surface and a secondary surface that is not based on three-dimensional beam-type elements, the first local tangent direction is tangential to the cross-section used to generate the analytical rigid surface, and the second local tangent direction is orthogonal to the plane of the cross-section in which the contact occurs.

In other cases, default initial orientations of the two local tangent directions are calculated by first computing tentative $\mathbf{t_1}$ and $\mathbf{t_2}$ directions. For element-based secondary surfaces the tentative directions are based on the secondary surface using the standard convention for calculating surface tangents. For node-based secondary surfaces the tentative $\mathbf{t_1}$ and $\mathbf{t_2}$ directions are set at each node to coincide with the global *x*- and *y*-axes, respectively. Abaqus constructs an orthogonal triad of $\mathbf{t_1}$, $\mathbf{t_2}$, and $\mathbf{n}$ (where $\mathbf{n} = \mathbf{t_1} \times \mathbf{t_2}$), then

rotates this triad such that $\mathbf{n}$ becomes aligned with the main surface normal at the tracked point on the main surface.

Small-sliding, surface-to-surface formulation

The default initial orientations of the two local tangent directions are based on the main surface normal, using the standard convention for calculating surface tangents.

Small-sliding, node-to-surface formulation

The default initial orientations of the two local tangent directions are calculated at each point on the main surface based on the main surface normal, using the standard convention for calculating surface tangents.

Defining Alternative Initial Local Tangent Directions for Contact Pair Surfaces

If the default local tangent directions are not convenient to prescribe an anisotropic friction model or to view contact output, you can define the local tangent directions for three-dimensional contact pair surfaces. You cannot redefine the local tangent directions for the following types of surfaces:

- Surfaces in a general contact domain
- Analytical rigid surfaces
- Two-dimensional surfaces

You define the local tangent directions by associating an orientation definition (see *Orientations*) with a contact pair surface. You can assign an orientation only to one surface of a contact pair. The surface on which an orientation can be defined is the same surface on which the default orientation would be calculated (see the conventions given previously). For example, an orientation can be defined only on the secondary surface in deformable versus deformable finite-sliding contact. If a second orientation is also given, an error message is issued. Therefore, it is not possible to redefine the local tangent directions for finite-sliding contact between a deformable secondary surface and an analytical rigid surface.

An orientation that is defined on a secondary surface of a contact pair that is generated from three-dimensional truss-type elements or from a list of nodes without rotational degree of freedoms will not be rotated if the secondary surface undergoes finite motion. In this case a warning message is issued during input processing.

Input File Usage: **CONTACT PAIR*, INTERACTION=*interaction_property_name*
secondary surface name, *main surface name*, *orientation for secondary surface*
secondary surface name, *main surface name*, , *orientation for main surface*

Abaqus/CAE Usage: You cannot define alternative local tangent directions for contact pairs in Abaqus/CAE.

Evolution of the Local Tangent Directions

For geometrically nonlinear analyses the local tangent directions rotate with the surface on which these directions were initially calculated or redefined using an orientation definition as described above with the exception that the local tangent direction rotates with the main surface for the small-sliding, surface-to-surface formulation. These rotated local tangent directions are further rotated to ensure that the normal vector, computed using the cross product of the rotated local tangent directions, corresponds to the normal vector on the main surface when the secondary node comes into contact.

Contact Constraint Enforcement Methods in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About General Contact in Abaqus/Standard*
- *About Contact Pairs in Abaqus/Standard*
- *About Mechanical Contact Properties*
- *Contact Pressure-Overclosure Relationships*
- **SURFACE BEHAVIOR*
- **CONTACT CONTROLS*
- **CONTACT CONTROLS ASSIGNMENT*
- *Defining general contact*
- *Defining surface-to-surface contact*
- *Defining a contact interaction property*

Overview

Contact constraint enforcement methods in Abaqus/Standard:

- are specified as part of the contact controls assignment definition (recommended) or surface interaction definition;
- determine how contact constraints imposed by a physical pressure-overclosure relationship (see *Contact Pressure-Overclosure Relationships*) are resolved numerically in an analysis;
- can either strictly enforce or approximate the physical pressure-overclosure relationships;
- can be modified to resolve convergence difficulties due to overconstraints; and
- sometimes utilize Lagrange multiplier degrees of freedom.

The available constraint enforcement methods for normal contact in Abaqus/Standard are discussed in detail in this section. The frictional constraint enforcement methods in Abaqus/Standard are assigned independently of those for the normal contact constraints and are discussed in *Frictional Behavior*. The use of Lagrange multipliers in contact calculations is also covered in this section.

Available Constraint Enforcement Methods in Abaqus/Standard

There are three contact constraint enforcement methods available in Abaqus/Standard:

- The direct method attempts to strictly enforce a given pressure-overclosure behavior per constraint, without approximation or use of augmentation iterations.
- The penalty method is a stiff approximation of hard contact.
- The augmented Lagrange method uses the same kind of stiff approximation as the penalty method, but also uses augmentation iterations to improve the accuracy of the approximation.

The default constraint enforcement method depends on interaction characteristics, as follows:

- The penalty method is used by default for finite-sliding, surface-to-surface contact if a “hard” pressure-overclosure relationship is in effect. For general contact, both the small- and finite-sliding tracking approaches use the penalty method by default.

- The augmented Lagrange method is used by default for three-dimensional self-contact with node-to-surface discretization if a “hard” pressure-overclosure relationship is in effect.
- The direct method is the default in all other cases.

You should consider the following factors when choosing the contact enforcement method:

- The direct method must be used for contact pairs with a “softened” pressure-overclosure relationship (see *Contact Pressure-Overclosure Relationships*).
- The direct method strictly enforces the specified pressure-overclosure behavior consistent with the constraint formulation
- The penalty or augmented Lagrange constraint enforcement methods sometimes provide more efficient solutions (generally due to reduced calculation costs per iteration and a lower number of overall iterations per analysis) at some (typically small) sacrifice in solution accuracy. See the discussions of the penalty and augmented Lagrange methods below.
- Overconstraints due to overlapping contact definitions or the combination of contact and other constraint types (see *Overconstraint Checks*) should be avoided for directly enforced hard contact.

You can define the enforcement method as follows:

- You can define surface pairwise enforcement through contact controls. This method is recommended.
- You can define enforcement through a surface interaction property.

  This is the only option available in Abaqus/CAE.

Direct Method

The direct method strictly enforces a given pressure-overclosure behavior for each constraint, without approximation or use of augmentation iterations.

Input File Usage: Use both of the following options:

**CONTACT*
**CONTACT CONTROLS ASSIGNMENT*, TYPE=ENFORCEMENT METHOD
surface_1, surface_2, DIRECT

Abaqus/CAE Usage: This method is not supported in Abaqus/CAE. Please refer to the alternative option to use Abaqus/CAE as discussed in *Direct Method*.

Direct Method for Hard Pressure-Overclosure Behavior

The direct method can be used to strictly enforce a “hard” pressure-overclosure relationship. Lagrange multipliers are always used in this case.

Direct Method for Softened Pressure-Overclosure Relationships

The direct method is the only method that can be used to enforce “softened” pressure-overclosure relationships. The direct method can be used to model softened contact behavior regardless of the type of contact formulation; however, modeling stiff interface behavior with a contact formulation that is prone to overconstraints can be difficult. Lagrange multipliers are used if the slope of the pressure-overclosure curve exceeds 1000 times the underlying element stiffness (as computed by Abaqus/Standard); otherwise, the constraints are enforced without Lagrange multipliers. The usage of Lagrange multipliers, thus, depends on the contact pressure. Softened pressure-overclosure relationships are discussed in more detail in *Contact Pressure-Overclosure Relationships*.

Limitations of the Direct Method

Because of its strict interpretation of contact constraints, hard contact simulations utilizing the direct enforcement method are susceptible to overconstraint issues. As a result, directly enforced hard contact is not available for contact pairs defined using three-dimensional self-contact with node-to-surface discretization. In this instance you can use an alternate enforcement method or the direct method with a softened pressure-overclosure relationship.

You might experience similar overconstraint problems with symmetric main-secondary contact pairs (see *Using Symmetric Main-Secondary Contact Pairs to Improve Contact Modeling*). Although directly enforced hard contact is the default for these contact pairs, it is recommended that you use an alternate enforcement method or a softened contact relationship.

Certain second-order element faces do not perform well in directly enforced hard contact relationships. See *Three-Dimensional Surfaces with Second-Order Faces and a Node-to-Surface Formulation* for details on this issue.

Penalty Method

The penalty method approximates hard pressure-overclosure behavior. With this method the contact force is proportional to the penetration distance, so some degree of penetration will occur. Advantages of the penalty method include:

- Numerical softening associated with the penalty method can mitigate overconstraint issues and reduce the number of iterations required in an analysis.
- The penalty method can be implemented such that no Lagrange multipliers are used, which allows for improved solver efficiency.

Choosing a Penalty Method

Abaqus/Standard offers linear and nonlinear variations of the penalty method. With the linear penalty method the so-called penalty stiffness is constant, so the pressure-overclosure relationship is linear. With the nonlinear penalty method the penalty stiffness increases linearly between regions of constant low initial stiffness and constant high final stiffness, resulting in a nonlinear pressure-overclosure relationship. The default penalty method is linear.

A comparison of the linear and nonlinear pressure-overclosure relationships with the default settings is shown in *Figure 1*.

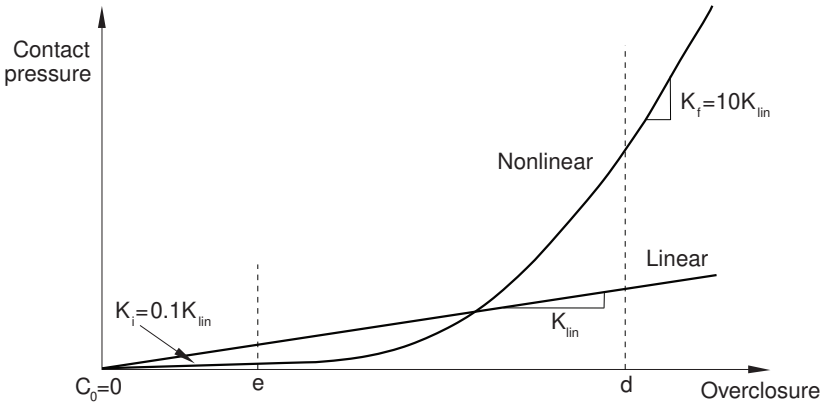


Figure 1: Comparison of linear and nonlinear pressure-overclosure relationships in a general analysis with default settings.

Linear Penalty Method

When the linear penalty method is used, Abaqus/Standard will, by default, set the penalty stiffness to 10 times a representative underlying element stiffness. You can scale or reassign the penalty stiffness, as discussed in *Modifying a Linear Penalty Stiffness* below. Contact penetrations resulting from the default penalty stiffness will not significantly affect the results in most cases; however, these penetrations can sometimes contribute to some degree of stress inaccuracy (for example, with displacement-controlled loading and a coarse mesh). The linear penalty method is used by default for the finite-sliding, surface-to-surface contact formulation. The linear penalty method is also the default for general contact, irrespective of whether the small- or finite-sliding tracking approach applies to contact constraints.

Input File Usage: Use both of the following options to specify the linear penalty method:

**CONTACT*
**CONTACT CONTROLS ASSIGNMENT*, TYPE=ENFORCEMENT METHOD
surface_1, surface_2, PENALTY

Abaqus/CAE Usage: Defining the linear penalty method through the recommended user interface is not supported in Abaqus/CAE. Please refer to the alternative option to use Abaqus/CAE as discussed in *Linear Penalty Method*.

Nonlinear Penalty Method

With the nonlinear penalty method, the pressure-overclosure curve has four distinct regions shown in *Figure 2*.

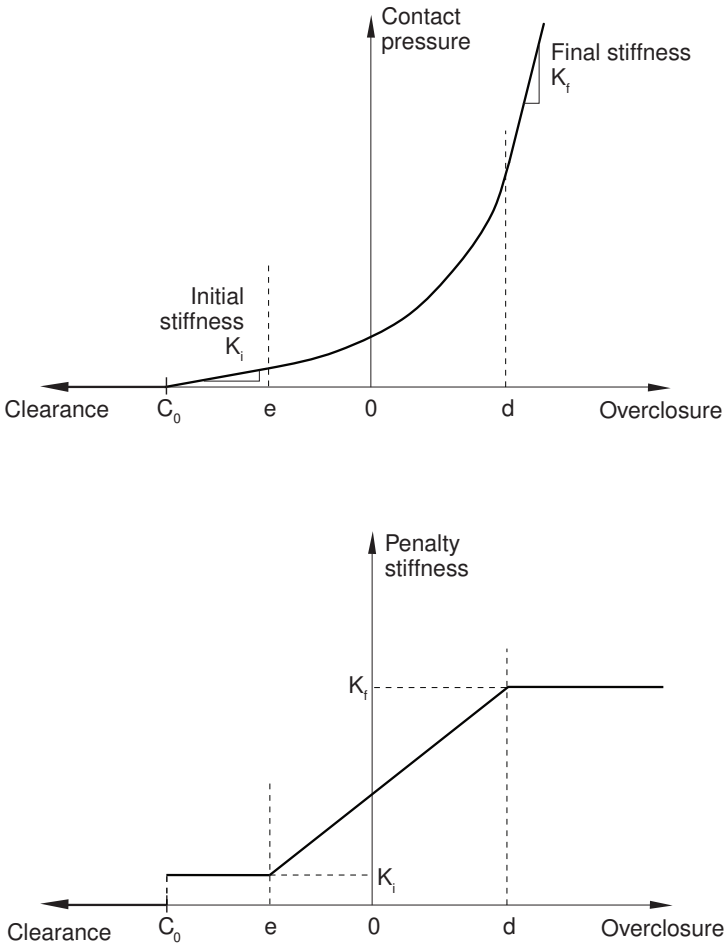


Figure 2: Nonlinear penalty pressure-overclosure relationship.

- Inactive contact regime: The contact pressure remains zero for clearances greater than c_0. The default setting of c_0 is zero.
- Constant initial penalty stiffness regime: The contact pressure varies linearly, with a slope equal to K_i for penetrations (overclosures) in the range $-c_0$ to e. The default initial penalty stiffness, K_i, is equal to the representative underlying element stiffness. The default value of e is 1% of a characteristic length computed by Abaqus/Standard to represent a typical facet size.
- Stiffening regime: The contact pressure varies quadratically for penetrations in the range e to d, while the penalty stiffness increases linearly from K_i to K_f. The default final penalty stiffness, K_f, is equal to 100 times the representative underlying element stiffness. The default value of d is 3% of the same characteristic length used to compute e (discussed above).
- Constant final penalty stiffness regime: The contact pressure varies linearly, with a slope equal to K_f for penetrations greater than d.

The low initial penalty stiffness typically results in better convergence of the Newton iterations and better robustness, while the higher final stiffness keeps the overclosure at an acceptable level as the contact pressure builds up. For linear perturbation procedures the default penalty stiffness is constant and equal to 100 times the representative underlying element stiffness, independent of the penalty stiffness used in the base state. Thus, it is equal to the constant final penalty stiffness.

Input File Usage: Use both of the following options to specify the linear penalty method:

**CONTACT*
**CONTACT CONTROLS ASSIGNMENT*, TYPE=ENFORCEMENT METHOD
surface_1, surface_2, NONLINEAR PENALTY

Abaqus/CAE Usage: Defining the nonlinear penalty method through the recommended user interface is not supported in Abaqus/CAE. Please refer to the alternative option to use Abaqus/CAE as discussed in *Nonlinear Penalty Method*.

Modifying the Penalty Stiffness

If you are interested in investigating the effects of modifying the penalty stiffness, it is generally recommended that you consider order-of-magnitude changes. Increasing the penalty stiffness above the threshold value discussed above will, by default, introduce Lagrange multipliers.

Modifying a Linear Penalty Stiffness

As part of the surface behavior definition, you can specify the linear penalty stiffness, shift the pressure-overclosure relationship by specifying the clearance at which the contact pressure is zero, or scale the default or specified penalty stiffness by a factor.

Input File Usage: To modify the linear penalty behavior in the contact controls assignment definition:

**CONTACT CONTROLS ASSIGNMENT*, TYPE=ENFORCEMENT METHOD
surface_1, surface_2, PENALTY, *penalty stiffness*, *clearance at zero pressure*, *factor*

Abaqus/CAE Usage: Defining linear penalty behavior through contact controls is not supported in Abaqus/CAE.

Modifying a Nonlinear Penalty Stiffness

As part of the surface behavior definition, you can specify the final nonlinear penalty stiffness, shift the pressure-overclosure relationship by specifying the clearance at which the contact pressure is zero, or scale the default or specified penalty stiffness by a factor. In addition, you can control directly the ratio of the initial to the final penalty stiffness, the scale factor, and the ratio that determines d and e.

Input File Usage: To modify the nonlinear penalty behavior in the contact controls assignment definition:

**CONTACT CONTROLS ASSIGNMENT*, TYPE=ENFORCEMENT METHOD
surface_1, surface_2, NONLINEAR PENALTY, *final penalty stiffness*, *clearance at zero pressure*, *factor*, *upper quadratic limit scale factor*, *ratio of initial penalty stiffness over final penalty stiffness*, *lower quadratic limit ratio*

Abaqus/CAE Usage: Defining nonlinear penalty behavior through contact controls is not supported in Abaqus/CAE.

Scaling the Penalty Stiffness on a Step-by-Step Basis

You can also scale the penalty stiffness on a step-by-step basis, which will act as an additional multiplier on any scale factor specified as part of the surface behavior definition.

Input File Usage: To scale the penalty stiffness on a step-by-step basis:

**CONTACT CONTROLS*, STIFFNESS SCALE FACTOR=*factor*

Abaqus/CAE Usage: To scale the penalty stiffness on a step-by-step basis:

Interaction module: Abaqus/Standard contact controls editor: **Augmented Lagrange**: **Stiffness scale factor:** *factor*

Adjusting the Penalty Stiffness across Iterations of the First Increment

It is common to have convergence difficulties in the first increment of an analysis if the contact status changes over a large portion of the contact area upon initial loading. An approach that tends to improve convergence behavior without sacrificing accuracy is to use a reduced penalty stiffness in the early iterations of the first increment and return to the default penalty stiffness for the final iterations of the first increment and all iterations of subsequent increments. Use of a reduced penalty stiffness in early iterations helps to robustly find an approximate contact status distribution, and the goal of later iterations is to then find an accurate solution, which is reported as the converged solution for the first increment.

Input File Usage: To scale the penalty stiffness within the first increment:

**CONTACT CONTROLS*, STIFFNESS SCALE FACTOR=USER ADAPTIVE

Limitations of the Penalty Method

The penalty method cannot be used for debonded surfaces.

If the penalty method is specified, Lagrange multipliers are always used during analysis steps with the following procedures:

- Design sensitivity analysis (see *Direct Design Sensitivity Analysis*)
- Direct steady-state dynamic analysis (see *Direct-Solution Steady-State Dynamic Analysis*)
- Quasi-Newton method (see *Convergence Criteria for Nonlinear Problems*)

If surface elements have been used to define a contact surface on the exterior of a substructure (see *Contact Modeling If Substructures Are Present*), Abaqus/Standard interprets the underlying element stiffness to be zero. This can lead to difficulty in determining the default penalty stiffness and may cause numerical problems during the analysis.

Augmented Lagrange Method

The linear penalty method can be used within an augmentation iteration scheme that drives down the penetration distance. This so-called augmented Lagrange method applies only to hard pressure-overclosure relationships. The following describes the sequence that occurs in each increment with this approach:

1. Abaqus/Standard finds a converged solution with the penalty method.
2. If a secondary node penetrates the main surface by more than a specified penetration tolerance, the contact pressure is “augmented” and another series of iterations is executed until convergence is once again achieved.

3. Abaqus/Standard continues to augment the contact pressure and find the corresponding converged solution until the actual penetration is less than the penetration tolerance.

The augmented Lagrange method might require additional iterations in some cases; however, this approach can make the resolution of contact conditions easier and avoid problems with overconstraints, while keeping penetrations small. The augmented Lagrange method is used by default for three-dimensional self-contact using node-to-surface discretization.

The default penetration tolerance is one-tenth of a percent of the characteristic interface length except in the following cases:

- if you specify a penalty stiffness scaling factor, f_k , of less than 1.0 (using the interface discussed below),

  Abaqus/Standard will automatically scale the default penetration tolerance by a factor of $\frac{1}{\sqrt{f_k}}$ (which will be greater than or equal to 1.0);
- the default penetration tolerance for finite-sliding, surface-to-surface contact is five percent of the characteristic interface length, subject to the scaling discussed in the previous bullet point.

The default penalty stiffness for the augmented Lagrange method is 1000 times the representative underlying element stiffness. Lagrange multipliers are used for the augmented Lagrange method if the penalty stiffness exceeds 1000 times the representative underlying element stiffness computed by Abaqus/Standard; otherwise, no Lagrange multipliers are used. Therefore, Lagrange multipliers are not used for the augmented Lagrange method with the default penalty stiffness.

Input File Usage: Use both of the following options to specify the linear penalty method:

**CONTACT*
**CONTACT CONTROLS ASSIGNMENT*, TYPE=ENFORCEMENT METHOD
surface_1, surface_2, AUGMENTED LAGRANGE

Abaqus/CAE Usage: Defining the augmented Lagrange method through contact controls is not supported in Abaqus/CAE.

Modifying the Penetration Tolerance for the Augmented Lagrange Method

You can modify the penetration tolerance for the augmented Lagrange method on a step-by-step basis by specifying an absolute or relative penetration tolerance. The relative penetration tolerance is specified with respect to a characteristic length computed by Abaqus/Standard. The default penetration tolerance was discussed above. The default penetration tolerance is increased automatically if you set the penalty stiffness scale factor to a value less than 1.0 (also discussed above); however, Abaqus/Standard will not adjust any directly specified penetration tolerance. Choosing a very small penetration tolerance might result in an excessive number of augmentation iterations.

Input File Usage: To specify an absolute penetration tolerance:

**CONTACT CONTROLS*, ABSOLUTE PENETRATION TOLERANCE=*tolerance*

To specify a relative penetration tolerance:

**CONTACT CONTROLS*, RELATIVE PENETRATION TOLERANCE=*tolerance*

Abaqus/CAE Usage: Interaction module: Abaqus/Standard contact controls editor: **Augmented Lagrange**: **Penetration tolerance: Absolute:** *tolerance* or **Relative:** *tolerance*

Modifying the Penalty Stiffness for the Augmented Lagrange Method

As with the penalty method, you can specify the penalty stiffness, shift the pressure-overclosure relationship by specifying the clearance at which the contact pressure is zero, or scale the default or specified penalty stiffness by a factor as part of the surface behavior definition. You can also scale the penalty stiffness on a step-by-step basis, which will act as an additional multiplier on any scale factor specified as part of the surface behavior definition. Choosing a very low penalty stiffness might result in an excessive number of augmentation iterations.

Input File Usage: To modify the penalty behavior in the contact controls assignment definition:

**CONTACT CONTROLS ASSIGNMENT*, TYPE=ENFORCEMENT METHOD
surface_1, surface_2, AUGMENTED LAGRANGE, *penalty stiffness*, *clearance at zero pressure*, *factor*

Abaqus/CAE Usage: Defining augmented Lagrange behavior through contact controls is not supported in Abaqus/CAE.

Modifying the Number of Allowed Augmentations for the Augmented Lagrange Method

You can define the number of allowed augmentations for the augmented Lagrange method.

Input File Usage: **CONTROLS*, PARAMETERS=TIME INCREMENTATION

, , , , , , , , , , , , I_A^c

Abaqus/CAE Usage: Defining the number of allowed augmentations for the augmented Lagrange method is not supported in Abaqus/CAE.

Limitations of the Augmented Lagrange Method

The augmented Lagrange method cannot be used for debonded surfaces.

If the augmented Lagrange method is specified, Lagrange multipliers are always used during analysis steps with the following procedures:

- Design sensitivity analysis (see *Direct Design Sensitivity Analysis*)
- Direct steady-state dynamic analysis (see *Direct-Solution Steady-State Dynamic Analysis*)
- Quasi-Newton method (see *Convergence Criteria for Nonlinear Problems*)

If surface elements have been used to define a contact surface on the exterior of a substructure (see *Contact Modeling If Substructures Are Present*), Abaqus/Standard interprets the underlying element stiffness to be zero. This can lead to difficulty in determining the default penalty stiffness and might cause numerical problems during the analysis.

Use of Lagrange Multiplier Degrees of Freedom by the Various Methods

Using Lagrange multipliers to enforce contact constraints can add significantly to the solution cost, but they also protect against numerical errors related to ill-conditioning that can occur if a high contact stiffness is in effect. Abaqus/Standard automatically chooses whether the constraint method makes use of Lagrange multipliers, based on a comparison of the contact stiffness to the underlying element stiffness. *Table 1* summarizes the use of Lagrange multipliers. Lagrange multipliers are not used for the default contact stiffnesses associated with the

penalty and augmented Lagrange approximations of hard contact. Any Lagrange multipliers associated with contact are present only for active contact constraints, so the number of equations might change as the contact status changes.

Table 1: Use of Lagrange multipliers in constraint enforcement methods.

Constraint Method	Use Lagrange Multipliers	
	Yes	No[1]
Direct, hard contact	Always	Never
Direct, exponential softened contact	If $k > 1000k_e$	If $k \leq 1000k_e$
Direct, linear softened contact	If $k > 1000k_e$	If $k \leq 1000k_e$
Direct, tabular softened contact	If $k > 1000k_e$	If $k \leq 1000k_e$
Penalty, hard contact	If $k_{penalty} > 1000k_e$	If $k_{penalty} \leq 1000k_e$
Augmented Lagrange, hard contact	If $k_{penalty} > 1000k_e$	If $k_{penalty} \leq 1000k_e$
k = slope of pressure-overclosure relationship		
$k_{penalty}$ = penalty stiffness		
k_e = underlying element stiffness		
[1]Lagrange multipliers are always used, regardless of the constraint enforcement method or stiffness, in the following cases: design sensitivity analyses, direct steady-state dynamics analyses, analyses using the quasi-Newton method.		

Alternative Constraint Enforcement Method

The alternative contact constraint enforcement method applies the enforcement method through a surface interaction definition. In this case any contact in the model associated with a particular interaction property will have the specified enforcement method. While this method is not preferred, it can be useful for cases where the physical contact properties and the best enforcement method closely align for the same regions of contact.

Direct Method

The direct method is discussed in more detail in *Direct Method*.

Input File Usage: Use both of the following options:

**SURFACE INTERACTION*, NAME=*interaction_property_name*
**SURFACE BEHAVIOR*, DIRECT

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical**->**Normal Behavior**: **Constraint enforcement method**: **Direct (Standard)**

Linear Penalty Method

The linear penalty method is discussed in more detail in *Penalty Method*.

Input File Usage: Use both of the following options to specify the linear penalty method:

*SURFACE INTERACTION, NAME=*interaction_property_name*
*SURFACE BEHAVIOR, PENALTY=LINEAR

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method**: **Penalty (Standard)**, **Behavior**: **Linear**

The linear penalty stiffness can be modified as discussed in more detail in *Modifying a Linear Penalty Stiffness*.

Input File Usage: To modify the linear penalty behavior in the surface behavior definition:

*SURFACE BEHAVIOR, PENALTY=LINEAR
penalty stiffness, *clearance at zero pressure*, *factor*

Abaqus/CAE Usage: To modify the linear penalty behavior in the surface behavior definition:

Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method: Penalty (Standard)**, **Behavior: Linear**, **Stiffness value: Specify:** *penalty stiffness*, **Stiffness scale factor:** *factor*, **Clearance at which contact pressure is zero:** *clearance at zero pressure*

Nonlinear Penalty Method

The nonlinear penalty method is discussed in more detail in *Penalty Method*.

Input File Usage: Use both of the following options to specify the nonlinear penalty method:

*SURFACE INTERACTION, NAME=*interaction_property_name*
*SURFACE BEHAVIOR, PENALTY=NONLINEAR

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method**: **Penalty (Standard)**, **Behavior**: **Nonlinear**

The nonlinear penalty stiffness can be modified as discussed in more detail in *Modifying a Nonlinear Penalty Stiffness*.

Input File Usage: To modify the nonlinear penalty behavior in the surface behavior definition:

*SURFACE BEHAVIOR, PENALTY=NONLINEAR
final penalty stiffness, *clearance at zero pressure*, *factor*, *upper quadratic limit scale factor*, *ratio of initial penalty stiffness over final penalty stiffness*, *lower quadratic limit ratio*

Abaqus/CAE Usage: To modify the nonlinear penalty behavior in the surface behavior definition:

Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method: Penalty (Standard)**, **Behavior: Nonlinear**, **Maximum stiffness value: Specify:** *final penalty stiffness*, **Stiffness scale factor:** *factor*, **Initial/Final stiffness ratio:** *ratio of initial penalty stiffness over final penalty stiffness*, **Upper quadratic limit scale factor:** *upper quadratic limit scale factor*, **Lower quadratic limit ratio:** *lower quadratic limit ratio*, **Clearance at which contact pressure is zero:** *clearance at zero pressure*

Augmented Lagrange Method

The augmented Lagrange method is discussed in more detail in *Augmented Lagrange Method*.

Input File Usage: Use both of the following options:

**SURFACE INTERACTION*, NAME=*interaction_property_name*
**SURFACE BEHAVIOR*, AUGMENTED LAGRANGE

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method: Augmented Lagrange (Standard)**

The penalty stiffness can be modified as discussed in more detail in *Modifying the Penalty Stiffness for the Augmented Lagrange Method*.

Input File Usage: To modify the penalty behavior in the surface behavior definition:

**SURFACE BEHAVIOR*, AUGMENTED LAGRANGE
penalty stiffness, *clearance at zero pressure*, *factor*

To scale the penalty stiffness on a step-by-step basis:

**CONTACT CONTROLS*, STIFFNESS SCALE FACTOR=*factor*

Abaqus/CAE Usage: To modify the penalty behavior in the surface behavior definition:

Interaction module: contact property editor: **Mechanical->Normal Behavior**: **Constraint enforcement method: Augmented Lagrange (Standard)**, **Stiffness value: Specify:** *penalty stiffness*, **Stiffness scale factor:** *factor*, **Clearance at which contact pressure is zero:** *clearance at zero pressure*

To scale the penalty stiffness on a step-by-step basis:

Interaction module: Abaqus/Standard contact controls editor: **Augmented Lagrange**: **Stiffness scale factor:** *factor*

Smoothing Contact Surfaces in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About General Contact in Abaqus/Standard*
- *About Contact Pairs in Abaqus/Standard*
- **CONTACT*
- **CONTACT PAIR*
- **SURFACE PROPERTY ASSIGNMENT*
- **SURFACE SMOOTHING*

Overview

With the finite element method, curved geometric surfaces are naturally approximated as a faceted group of connected element faces. This section discusses methods to improve faceted surface representations for purposes of contact computations based on knowledge of idealized initial surface geometry. Other types of surface smoothing are discussed in *Smoothing Main Surfaces for the Finite-Sliding, Node-to-Surface Formulation* and *Using the Small-Sliding Tracking Approach*.

The use of a faceted surface geometry rather than the true surface geometry can significantly contribute to contact stress inaccuracy in contact interactions, especially when the magnitude of the differences between the faceted and true surface is not small with respect to the deformation of the components in contact. Contact stress output is of primary importance in many applications; for example, the distribution of contact pressures can be used to identify wear patterns and peak pressure values to determine relative lives of machine parts.

Abaqus/Standard offers techniques to improve the accuracy and robustness of contact computations based on comparisons between the initial faceted geometry and a more idealized initial geometry of the same surface. In some cases you may know that the idealized surface is (exactly or approximately) cylindrical, spherical, or toroidal. Abaqus/CAE identifies cylindrical, spherical, and toroidal surfaces automatically. When creating finite element models as CAD representations in the **3D**EXPERIENCE platform, idealized surface representations are often available for selection.

Smoothing of Common Curved Surface Geometries

One method of surface smoothing applies to surface regions that are roughly axisymmetric, roughly spherical, or part of a toroidal surface. The smoothing method applies to general contact and surface-to-surface contact pairs. For example, the pin insertion model in *Figure 1* could benefit from this smoothing: the body of the pin is cylindrical, the head of the pin is hemispherical, and the hole is conical. Surface-to-surface contact smoothing would also be effective if the surfaces were not perfectly axisymmetric, spherical, or toroidal; for example, if the pin body were slightly elliptical.

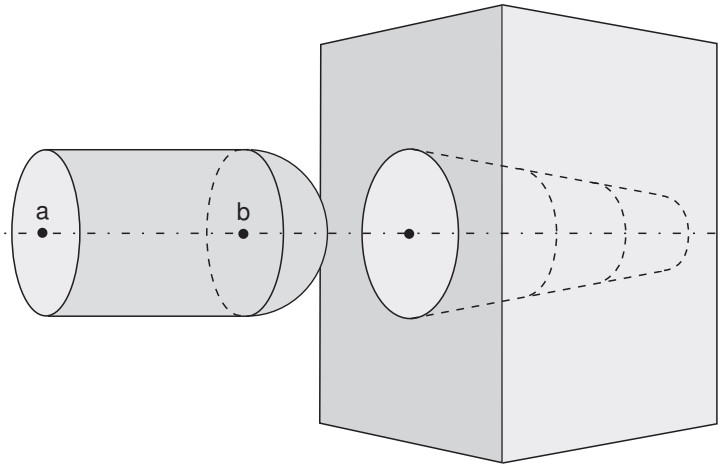


Figure 1: Surface-to-surface contact model with surface smoothing.

Effects of Contact Surface Smoothing

The impact of contact smoothing based on comparison of initial faceted surface geometry to initial idealized surface geometry can be demonstrated by a simple model. *Figure 2* shows the initial mesh geometry for a two-dimensional model of concentric cylinders with an interference fit. The concentric cylinders are modeled with first-order elements of different sizes.

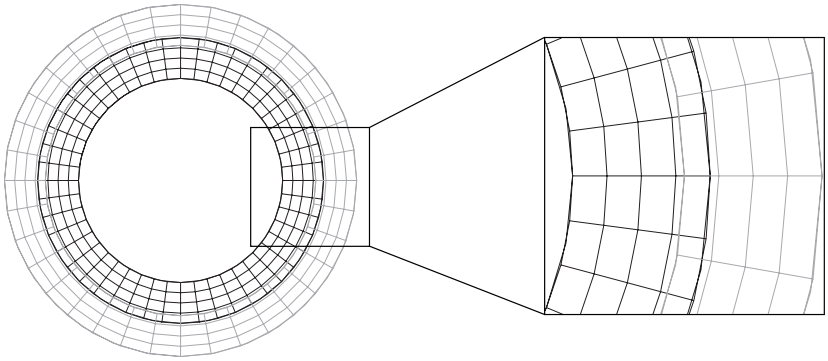


Figure 2: Initial mesh geometry for interference fit model.

Discrepancies between the true surface geometry and the faceted surface geometry result in noise in the contact pressure solution if surface smoothing is not used. If the interference distance and resulting deformation distance is small with respect to the geometry discrepancy, this noise can have a significant effect on the accuracy of the solution. Although surface-to-surface contact typically handles these discrepancies better than node-to-surface contact, it is not unusual for the maximum deviation from the analytical pressure solution to be upward of 100%. The effects of the noise become less apparent for larger deformations, but they are never completely eliminated.

Applying Smoothing of Common Curved Surface Geometries to General Contact

Smoothing of common curved surface geometries for general contact is enabled by surface property assignments. Surface property assignments specify which surfaces are to be smoothed and the smoothing method to be used. The underlying geometry correction methods are the same for general contact and contact pairs:

- The circumferential smoothing method is applicable to surfaces approximating a portion of a circle in two dimensions or a portion of a surface of revolution in three dimensions.

- The spherical smoothing method is applicable to surfaces approximating a portion of a sphere in three dimensions.
- The toroidal smoothing method is applicable to surfaces approximating a portion of a torus in three dimensions (i.e., a circular arc revolved about an axis).

For each surface, you must specify the appropriate geometry correction method and either the approximate axis of revolution (for circumferential or toroidal smoothing) or the approximate spherical center (for spherical smoothing). For toroidal smoothing, you must also specify the distance of the center of the circular arc from the axis of revolution, and the line joining point (X_a, Y_a, Z_a) and the center of the circular arc should be perpendicular to the axis of revolution.

Input File Usage: Use the following option to apply a geometric correction and define the parameters by specifying coordinates:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=GEOMETRIC CORRECTION, DEFINITION=COORDINATES
data lines to define smoothing regions

Several examples for data line entries are shown below. Repeat the data lines as many times as necessary to define the appropriate geometry corrections for all surfaces in the contact domain.

To apply circumferential smoothing to a surface with an axis of symmetry passing through points (X_a, Y_a, Z_a) and (X_b, Y_b, Z_b):

surface, CIRCUMFERENTIAL, X_a, Y_a, Z_a, X_b, Y_b, Z_b

To apply spherical smoothing to a surface with a spherical center at point (X_a, Y_a, Z_a):

surface, SPHERICAL, X_a, Y_a, Z_a

To apply toroidal smoothing to a surface with an axis of symmetry passing through points (X_a, Y_a, Z_a) and (X_b, Y_b, Z_b) with the center of the revolved circular arc at a distance *R* from the axis of symmetry:

surface, TOROIDAL, X_a, Y_a, Z_a, X_b, Y_b, Z_b, R

Use the following option to apply a geometric correction and define the parameters by specifying nodes:

**SURFACE PROPERTY ASSIGNMENT*, PROPERTY=GEOMETRIC CORRECTION, DEFINITION=NODES
data lines to define smoothing regions

Several examples for data line entries are shown below. Repeat the data lines as many times as necessary to define the appropriate geometry corrections for all surfaces in the contact domain.

To apply circumferential smoothing to a surface with an axis of symmetry passing through nodes a and b:

surface, CIRCUMFERENTIAL, node a, node b

To apply spherical smoothing to a surface with a spherical center at node a:

surface, SPHERICAL, node a

To apply toroidal smoothing to a surface with an axis of symmetry passing through nodes a and b with the center of the revolved circular arc at a distance *R* from the axis of symmetry:

surface, TOROIDAL, node a, node b, R

Abaqus/CAE Usage: Contact surface smoothing can be applied only to native geometry models in Abaqus/CAE. By default, Abaqus/CAE automatically detects all circumferential, spherical, and toroidal surfaces in the general contact domain that can be smoothed and applies the appropriate smoothing.

Use the following option to prevent automatic surface smoothing of a model:

Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Surface smoothing assignments: Edit**:
toggle off **Automatically assign smoothing for geometric faces**

Use the following option to manually apply smoothing to a surface:

Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Surface smoothing assignments: Edit**:
Select surface, click the arrows to transfer surface to list of smoothing assignments.
In the **Smoothing Option** column, select **REVOLUTION** to apply circumferential smoothing, select **SPHERICAL** to apply spherical smoothing, select **TOROIDAL** to apply toroidal smoothing, or select **NONE** to prevent smoothing of the surface.

Example: Pin-in-Hole with General Contact

To improve contact pressure accuracy for the model in *Figure 1*, contact smoothing can be applied to both the main and secondary surfaces. Two different geometric correction methods are required for the pin (the secondary surface), so additional surfaces are defined corresponding to regions of the secondary surface. Spherical smoothing is defined for the tip of the pin. Since the body of the pin and the hole share an axis of revolution, circumferential smoothing is applied to both of these surfaces. This surface smoothing definition applies even if the cross-sectional shapes of the pin and hole deviate from perfect circles.

```
*SURFACE INTERACTION, NAME=FRICTION1
*CONTACT
*CONTACT INCLUSIONS
PIN, HOLE
*CONTACT PROPERTY ASSIGNMENT
 , , FRICTION1
*SURFACE PROPERTY ASSIGNMENT, PROPERTY=GEOMETRIC CORRECTION,
DEFINITION=COORDINATES
PIN_TIP, SPHERICAL, Xb, Yb, Zb
PIN_BODY, CIRCUMFERENTIAL, Xa, Ya, Za, Xb, Yb, Zb
HOLE, CIRCUMFERENTIAL, Xa, Ya, Za, Xb, Yb, Zb
```

Applying Smoothing of Common Curved Surface Geometries to Surface-to-Surface Contact Pairs

Smoothing of common curved surface geometries for contact pairs that use a surface-to-surface contact formulation is enabled by creating a surface smoothing definition. A contact pair definition references this smoothing definition to apply geometric corrections in the contact formulation (the physical geometry of the model is not altered).

The surface smoothing definition lists all of the faceted regions in the contact pair surfaces that must be smoothed, as well as the geometry correction method that should be applied to each region. Three geometry correction methods can be employed:

- The circumferential smoothing method is applicable to surfaces approximating a portion of a circle in two dimensions or a portion of a surface of revolution in three dimensions.
- The spherical smoothing method is applicable to surfaces approximating a portion of a sphere in three dimensions.
- The toroidal smoothing method is applicable to surfaces approximating a portion of a torus in three dimensions (i.e., a circular arc revolved about an axis).

Each surface-to-surface contact pair refers to a single smoothing definition; therefore, a smoothing definition must list all of the smoothed regions and applicable geometry correction methods for the contact pair. Geometry corrections can be applied to main surfaces and to secondary surfaces; you can also apply corrections to selected regions of each surface. A surface smoothing definition can include multiple regions and different geometric correction methods for each region. For each region, you must specify the appropriate geometry correction method and either the approximate axis of revolution (for circumferential or toroidal smoothing) or the approximate spherical center (for spherical smoothing). For toroidal smoothing, you must also specify the distance of the center of the circular arc from the axis of revolution, and the line joining point (X_a, Y_a, Z_a) and the center of the circular arc should be perpendicular to the axis of revolution.

Input File Usage: Use both of the following options to apply surface-to-surface contact smoothing:

**CONTACT PAIR*, GEOMETRIC CORRECTION=*smoothing_name*
**SURFACE SMOOTHING*, NAME=*smoothing_name*
data lines to define smoothing regions (see below)

Use the following data line to apply circumferential smoothing to surface regions with an axis of symmetry passing through points (X_a, Y_a, Z_a) and (X_b, Y_b, Z_b):

secondary_region, *main_region*, CIRCUMFERENTIAL, X_a, Y_a, Z_a, X_b, Y_b, Z_b

Use the following data line to apply spherical smoothing to surface regions with a spherical center at point (X_a, Y_a, Z_a):

secondary_region, *main_region*, SPHERICAL, X_a, Y_a, Z_a

Use the following data line to apply toroidal smoothing to surface regions with an axis of symmetry passing through points (X_a, Y_a, Z_a) and (X_b, Y_b, Z_b) with the center of the revolved circular arc at a distance R from the axis of symmetry:

secondary_region, *main_region*, TOROIDAL, X_a, Y_a, Z_a, X_b, Y_b, Z_b, R

Repeat the data lines as many times as necessary to define the appropriate geometry corrections for all surfaces in the contact pair.

Abaqus/CAE Usage: Abaqus/CAE can automatically identify any circumferential, spherical, or toroidal surfaces in a contact interaction that will benefit from contact smoothing and apply the necessary geometry correction methods.

Interaction module: contact interaction editor: **Surface Smoothing**: **Automatically smooth 3D geometry surfaces when applicable**

Surface-to-surface contact smoothing cannot be applied to surfaces on orphan mesh models.

Example: Pin-in-Hole with Contact Pairs

To improve contact pressure accuracy for the model in *Figure 1*, contact smoothing can be applied to both the main and secondary surfaces. Two different geometric correction methods are required for the pin (the secondary surface), so additional surfaces are defined corresponding to regions of the secondary surface. Spherical smoothing is defined for the tip of the pin. Since the body of the pin and the hole share an axis of revolution, circumferential smoothing is applied to both of these surfaces. This surface smoothing definition applies even if the cross-sectional shapes of the pin and hole deviate from perfect circles.

```
*CONTACT PAIR, TYPE=SURFACE TO SURFACE, INTERACTION=FRICTION1,
   GEOMETRIC CORRECTION=SMOOTH1
PIN, HOLE
*SURFACE INTERACTION, NAME=FRICTION1
*SURFACE SMOOTHING, NAME=SMOOTH1
PIN_TIP, , SPHERICAL, X_b, Y_b, Z_b
PIN_BODY, HOLE, CIRCUMFERENTIAL, X_a, Y_a, Z_a, X_b, Y_b, Z_b
```

Smoothing of Faceted Surfaces Based on Comparison to CAD Representations

Simulations using general contact submitted from **3D**EXPERIENCE platform Scenario Creation apps can use a technique to improve finite element representations of contact surfaces. This technique is based on deviations of the original element-based surface geometry from a more accurate reference representation of the undeformed geometry and leverages CAD surface representations available in the **3D**EXPERIENCE platform in the computation of contact geometry corrections. This technique is not available in explicit dynamic steps.

The CAD-based smoothing method is applicable to arbitrary surface shapes having CAD representations in the **3D**EXPERIENCE platform. Consider the example shown in *Figure 3*. This example is constructed such that the analytical solution has a uniform, uniaxial stress state of 10^8 Pa. The curved contact interface has closed contact and sticking friction everywhere, and the same elastic material exists on both sides of the contact interface. The meshes used in this example are made of quadratic hexahedral and wedge elements. The meshes do not have matched nodes across the contact interface, which causes significant stress noise if the contact surfaces are not smoothed, especially for small strains where the level of element deformation is less than or equal to the distances associated with the mismatch of faceted surface representations across the interface.

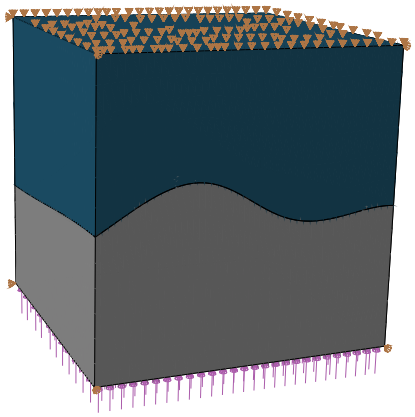


Figure 3: Two solid blocks contacting on an arbitrarily curved interface (subjected to uniform unidirectional compressive stress).

Figure 4 shows that CAD-enhanced contact reduces stress noise for this example. For example, the maximum Mises stress deviates from the analytical Mises stress solution by 33% without CAD-enhanced contact, and this deviation is less than 3% with CAD-enhanced contact. The CAD-based smoothing also significantly improves contact pressure results for this model.

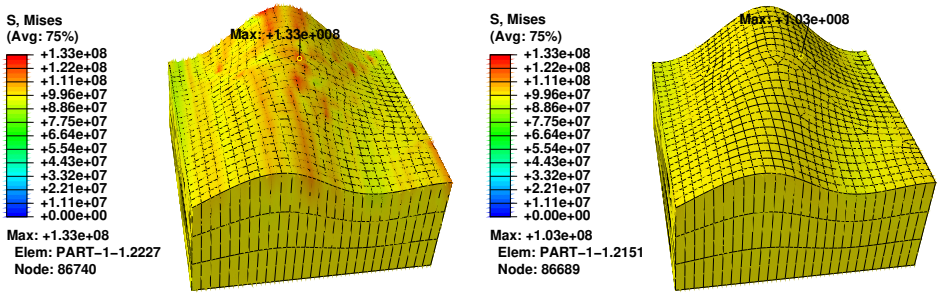


Figure 4: Comparison of the Mises stress field between the case without geometry-based corrections (left) and with geometry-based corrections applied to the curved contacting surfaces (right).

Considerations for Smoothing Methods Based on Knowledge of Idealized Initial Geometry

The smoothing methods in this section are used to adjust contact penetration/gap computations based on differences in undeformed geometries between faceted finite element surface representations and more idealized surface representations. The effects of the surface smoothing techniques discussed in this section tend to be most significant for analyses involving small deformation and coarse mesh discretization with first-order elements in the contact region; however, significant improvements to contact stress solutions are common even when the mesh is quite refined or higher-order elements are used. For analyses with large deformation these smoothing techniques typically have an insignificant effect on solutions. However, in some cases the smoothing can degrade the solution accuracy after large deformation; therefore, it is not recommended to use these smoothing techniques for large-deformation analyses. The effectiveness of the smoothing does not degrade upon relative motion between contact surfaces; for example, the smoothing technique works well for cases involving large sliding but small deformation.

The surface smoothing techniques discussed in this section assume that initial locations of surface nodes lie on the true initial surface geometry, with the exception of midside nodes of higher-order elements. These smoothing techniques remain effective even if the midside nodes of higher-order elements do not lie on the true initial geometry (models meshed using Abaqus/CAE always have midside nodes placed on the true initial geometry, but this may not be the case with other meshing preprocessors).

Initial Contact Stresses in Abaqus/Standard

Products: Abaqus/Standard

References:

- *Initial Conditions*

Overview

Initial contact stresses are internally computed from user-specified initial element stresses.

Initial contact stresses provide nonzero initial conditions for contact in the first step of an Abaqus/Standard analysis. Initial element stresses can be specified directly in an input file, can be read from a prior analysis, or specified via a user subroutine. Based on these initial element stresses in the elements underlying the contact surfaces, initial contact stresses are computed automatically.

Initial Stresses for Contact

Initial stresses for contact are derived from initial stresses that you specify for elements underlying contact surfaces. This involves estimating both normal and shear contact traction such that they are approximately in equilibrium with the element initial stresses. The procedure for estimating initial stresses matches the true distribution very well for cases where underlying element stress distributions result in either uniform or linearly varying contact stresses. For example, linearly varying underlying element stresses result in linearly varying contact stresses on flat contact surfaces. For more complex spatial variations of contact stress, the initial contact stress distribution may only be approximate. In situations where the underlying element stress distribution is an equilibrium stress distribution, nonzero initial contact stresses are meant to ensure that the overall system that includes both contact and user elements is close to an overall equilibrium state with respect to initial loads and boundary conditions. Consequently, an initial static or geostatic equilibration step with no changes to loads and boundary conditions is likely to converge more easily.

In cases where initial element stresses are different across the contact interface, the secondary side of the element stresses is used to estimate the initial contact stresses. These stresses can be visualized in the zero-increment frame of an analysis or after a data check analysis. You can control whether initial contact stresses are to be calculated and reported.

Input File Usage: Use the following option to turn on initial contact stress calculations from element stresses:

**INITIAL CONDITIONS*, TYPE=STRESS, CORRESPONDING CONTACT STRESS=YES (default)
data lines to specify initial stresses

Use the following option to turn off initial contact stress calculations from element stresses:

**INITIAL CONDITIONS*, TYPE=STRESS, CORRESPONDING CONTACT STRESS=NO
data lines to specify initial stresses

Effect of Initial Contact Status on Initial Contact Stress Calculations

Initial contact stresses are calculated only when the initial contact status is closed. In other cases, contact stresses are reported as zero. It is strongly recommended that you use nodal adjustments (see *Contact Initialization for General Contact in Abaqus/Standard* and *Contact Initialization for Contact Pairs in Abaqus/Standard*) to set the initial contact status to closed. In the absence of nodal adjustments, even small variations in the contact gap along contact surfaces can lead to a very noisy distribution of initial contact status and, subsequently, the initial contact stress distribution. Such a noisy initial contact stress distribution may adversely affect convergence.

Dependence of Initial Contact Stresses on Mechanical Contact Properties

Initial contact stresses are calculated to be consistent with the mechanical contact properties that you specify for a particular contact region (see *About Mechanical Contact Properties*). For example, in the case of Coulomb friction the initial contact shear stresses are always capped at the maximum shear stresses that the interface can support based on initial contact normal stresses and the friction coefficient specified in the model definition. In situations where friction properties are changed during the first step (see *Frictional Behavior*), the friction specification at the model level prevails for initial stress calculations. Similarly, if the contact interface is specified to be frictionless, the initial contact shear stresses are reported as zero even if the underlying element stress distribution may result in nonzero shear tractions to satisfy equilibrium. For normal contact behavior, both tensile and compressive initial contact pressures are permitted if the interface is tied (see *Defining Tied Contact in Abaqus/Standard*) or no separation (see *Contact Pressure-Overclosure Relationships*) between contact surfaces is permitted once contact is established. However, for regular contact only compressive initial contact pressures are initialized. When cohesive behavior (see *Contact Cohesive Behavior*) or debonding (see *Crack Propagation Analysis*) is specified at the contact interface, initial contact stresses are not supported.

Interpreting Initial Contact Stresses with Pore Pressure Degrees of Freedom

In models with pore pressure degrees of freedom, the initial contact pressures are affected by pore pressure values depending on the specification of pore fluid flow (see *Pore Fluid Contact Properties*) at the contact interface. If pore fluid flow is allowed across the interface, contact normal pressures are initialized to their effective stress values. However, if there is no pore fluid flow across the interface, contact normal pressures are initialized to their total stress values. The latter may also happen in situations where a contact surface on one side of the contact interface has pore pressure degrees of freedom while the opposing surface does not.

Generally Applicable Contact Controls in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About Contact Pairs in Abaqus/Standard*
- **CONTACT CONTROLS*
- **CONTACT PAIR*
- *Defining surface-to-surface contact*
- *Defining self-contact*
- *Specifying contact controls in an Abaqus/Standard analysis*

Overview

Contact controls in Abaqus/Standard:

- should not be modified from the default settings for the majority of problems;
- can be used for problems where the standard contact controls do not provide cost-effective solutions;
- can be used for problems where the standard controls do not effectively establish the desired contact conditions; and
- can be used in some situations to control whether supplementary contact constraints are created.

Problems that benefit from adjustments to the contact controls in Abaqus/Standard are generally large models with complicated geometries and numerous contact interfaces.

Applying Contact Controls

You can apply contact controls on a step-by-step basis to all of general contact, contact pairs, and contact elements that are active in the step or to individual contact pairs. This makes it possible to apply contact controls to a specific contact pair to take the simulation through a difficult phase. Contact controls remain in effect until they are either changed or reset to their default values. If in any given step the contact controls are declared for both the entire model and for a specific contact pair, the controls for the specific contact pair will override those for the entire model for that contact pair.

In addition, you can specify supplementary contact constraints on individual contact pairs as described below in *Supplementary Contact Constraints*.

Input File Usage: To apply contact controls to all contact pairs and contact elements:

**CONTACT CONTROLS*
contact control options

To apply contact controls to a specific contact pair:

**CONTACT CONTROLS*, SECONDARY=*secondary surface*, MAIN=*main surface*
contact control options

Repeat this option to apply contact controls to several contact pairs.

Abaqus/CAE Usage: Contact controls in Abaqus/CAE can be applied only to specific contact pairs:

Interaction module: **Interaction->Contact Controls->Create**:
Abaqus/Standard contact controls
Contact interaction editor: **Contact controls:** *contact controls name*

Resetting Contact Controls

You can reset all contact controls to their default values, or you can reset the controls for a specific contact pair.

Input File Usage: To reset all contact controls:

**CONTACT CONTROLS*, RESET

To reset the controls for a specific contact pair:

**CONTACT CONTROLS*, SECONDARY=*secondary surface*,
MAIN=*main surface*, RESET

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Contact controls: (Default)**

You cannot reset all contact controls at once in Abaqus/CAE.

Automatic Stabilization of Rigid Body Motions in Contact Problems

Abaqus/Standard offers contact stabilization to help automatically control rigid body motion in static problems before contact closure and friction restrain such motion.

It is recommended that you first try to stabilize rigid body motion through modeling techniques (modifying geometry, imposing boundary conditions, etc.). The automatic stabilization capability is meant to be used in cases in which it is clear that contact will be established, but the exact positioning of multiple bodies is difficult during modeling. It is not meant to simulate general rigid body dynamics; nor is it meant for contact chattering situations or to resolve initially tight clearances between mating surfaces.

When automatic contact stabilization is used, Abaqus/Standard activates viscous damping for relative motions of the contact pair at all secondary nodes, in the same manner as contact damping (see *Contact Damping*). Unlike most contact controls, which carry over to subsequent steps until they are modified or reset, automatic stabilization damping is applied only for the duration of the step in which it is specified. In subsequent steps the stabilization is removed, even if contact was not established or if rigid body motions appear later because of complete separation of the contact pair. If needed, you should specify stabilization for subsequent steps as well.

By default, the damping coefficient:

- is calculated automatically for each contact constraint based on the stiffness of the underlying elements and the step time,
- is applied to all contact pairs equally in the normal and tangential directions,
- is ramped down linearly over the step,
- is active only when the distance between the contact surfaces is smaller than a characteristic surface dimension, and
- is zero for contact modeled with contact elements (such as gap contact elements, tube-to-tube contact elements, etc.).

Although the automatically calculated damping coefficient typically provides enough damping to eliminate the rigid body modes without having a major effect on the solution, there is no guarantee that the value is optimal or even suitable. This is particularly true for thin shell models, in which the damping may be too high. Hence, you may have to increase the damping if the convergence behavior is problematic or decrease the damping if it distorts the solution. The first case is obvious, but the latter case requires a postanalysis check. There are several ways to carry out such checks. The simplest method is to consider the ratio between the energy dissipated by viscous damping and a more general energy measure for the model, such as the elastic strain energy. These quantities can be obtained as output variables ALLSD and ALLSE, respectively. More detailed information can be obtained by comparing the contact damping stresses CDSTRESS (with the individual components CDPRESS, CDSHEAR1, and CDSHEAR2) to the true contact stresses CSTRESS (with the individual components CPRESS, CSHEAR1, and CSHEAR2). If the contact damping stresses are too high, you should decrease the damping. The comparison should be made after contact is firmly established; the contact damping stresses will always be relatively high when contact is not yet or only partially established.

The easiest way to increase or decrease the amount of damping is to specify a factor by which the automatically calculated damping coefficient will be multiplied. Typically, you should initially consider changing the default damping by (at least) an order of magnitude; if that addresses the problem sufficiently, you can do some subsequent fine-tuning. In some cases a larger or smaller factor may be needed; this is not a problem as long as a converged solution is obtained and the dissipated energy and contact damping stresses are sufficiently small.

It is also possible to specify the damping coefficient directly. Direct specification of the damping value is not easy and may require some trial and error. For efficiency reasons this may best be done on a similar model of reduced size. If the damping coefficient is specified directly, any multiplication factor specified for the default damping coefficient is ignored.

Input File Usage: To use the default damping coefficient:

**CONTACT CONTROLS*, STABILIZE

To specify a scale factor for the default damping coefficient:

**CONTACT CONTROLS*, STABILIZE=*factor*

To specify the damping coefficient directly:

**CONTACT CONTROLS*, STABILIZE
damping coefficient

Abaqus/CAE Usage: Interaction module: Abaqus/Standard contact controls editor: **Stabilization**: **Automatic stabilization**, **Factor:** *factor* or **Stabilization coefficient:** *damping coefficient*

Changing the Stabilization within Increments

To reduce or eliminate the likelihood of contact stabilization significantly influencing the reported solution, scale factors can be introduced that vary across iterations of an increment. Having more stabilization in effect during the early iterations of an increment can be helpful to avoid numerical problems prior to establishing some contact. Having less or no stabilization in effect during the later iterations can be helpful to improve the accuracy of the final converged iteration of an increment.

You can specify these scale factors. For example, specifying “1,0” results in the scale factor being unity during initial iterations (until various convergence measures are satisfied or nearly satisfied) and then the scale factor being reset to zero (effectively turning off stabilization) for the final iterations until convergence checks are again satisfied.

Input File Usage: To change the damping coefficients within an increment:

**CONTACT CONTROLS*, STABILIZE=USER ADAPTIVE

Specifying the Stabilization Ramp-Down Factor

You can specify the ramp-down factor at the end of the step. By default, this value is equal to zero, so that the damping vanishes completely at the end of the step. Entering a nonzero value for this factor can be useful in cases where the rigid body modes are not fully constrained at the end of the step; for example, if the problem is frictionless and sliding motions can occur but there is no net force in the sliding direction. In that case it is usually desirable to maintain the small damping in the next step by using the value used for the ramp-down as the multiplication factor for the damping coefficient. If needed, you can maintain this damping level by setting the ramp-down factor equal to one.

Input File Usage: **CONTACT CONTROLS*, STABILIZE
, *ramp-down factor*

Abaqus/CAE Usage: Interaction module: Abaqus/Standard contact controls editor: **Stabilization**: **Automatic stabilization** or **Stabilization coefficient**, **Fraction of damping at end of step:** *ramp-down factor*

Specifying the Damping Range

By default, the opening distance over which the damping is applied (the damping range) is equal to the characteristic secondary surface facet dimension; if such a dimension is not available (for example, in the case of a node-based surface), a characteristic element length obtained for the whole model is used. The damping is 100% of the reference value for openings less than half the damping range and from there is ramped to zero for an opening equal to the damping range. Alternatively, you can specify the damping range directly, overriding the calculated value. This can be useful if the damping should work only for a narrow gap, or if the damping should be in effect regardless of the opening distance. In the latter case a large value should be entered.

Input File Usage: **CONTACT CONTROLS*, STABILIZE
, , *damping range*

Abaqus/CAE Usage: Interaction module: Abaqus/Standard contact controls editor: **Stabilization**: **Automatic stabilization** or **Stabilization coefficient**, **Clearance at which damping becomes zero: Specify:** *damping range*

Specifying Tangential Damping

By default, the damping in the tangential direction is the same as the damping in the normal direction. However, if a lower or higher value is desired, you can decrease or increase the tangential damping or set it to zero.

Input File Usage: **CONTACT CONTROLS*, STABILIZE, TANGENT FRACTION=*value*

Abaqus/CAE Usage: Interaction module: Abaqus/Standard contact controls editor: **Stabilization**: **Automatic stabilization** or **Stabilization coefficient**, **Tangent fraction:** *value*

Contact Controls Associated with Normal Contact Constraints

These controls allow you to specify that nodes on the contact interfaces can violate "hard" contact conditions. In addition, these controls can be used to modify the behavior of the "softened" pressure-overclosure relationships and the augmented Lagrangian or penalty contact constraint enforcement. The no separation pressure-overclosure relationships cannot be modified by the contact controls.

A node can violate the contact condition in one of two ways. First, Abaqus/Standard may consider that there is no contact at that node, even though the node has penetrated the main surface by a small distance. Second, Abaqus/Standard may consider that there is contact at a node, even though the normal pressure transmitted between the contacting surfaces at the node is negative (that is, a tensile stress is being transmitted).

Modifying the Behavior of the Augmented Lagrangian or Penalty Contact Constraint Enforcement

For augmented Lagrangian contact you can specify the allowable penetration (either directly or as a fraction of a characteristic contact surface dimension) that is permitted to violate the impenetrability condition. In addition, for augmented Lagrangian or penalty contact you can scale the default penalty stiffness calculated by Abaqus/Standard. Controls for the augmented Lagrange and penalty constraint enforcement methods are discussed in *Contact Constraint Enforcement Methods in Abaqus/Standard*.

Modifying the Tangential Penalty Stiffness in Linear Perturbation Steps

The penalty stiffness used to enforce tangential constraints in linear perturbation steps generally differs from the penalty stiffness used to enforce sticking in a general step. In perturbation steps Abaqus/Standard activates the tangential contact constraints when the corresponding normal constraint is active in the base state and the contact property (surface interaction) definition includes a friction model. By default, the tangential penalty stiffness is equal to the default normal penalty stiffness.

You can scale the tangential penalty stiffness to simulate sticking/slipping conditions on a step-by-step basis. This scaling only affects the perturbation step in which it is specified; it will not carry over to subsequent steps. If you want the same scale factor applied in a series of perturbation steps, you must specify the scale factor explicitly in each step.

Some procedures that rely on a frequency analysis, such as complex frequency analysis and subspace-based steady-state dynamic analysis, are influenced by the scaling of the tangential stiffness that was in effect for the prior frequency analysis and the scaling of the tangential stiffness that is in effect for these steps. In such cases consistent scaling is recommended for these steps. For other mode-based procedures based on a frequency analysis, the scaling of the tangential stiffness is ignored and only the effect of the previous frequency analysis is considered.

Input File Usage: To modify the tangential penalty stiffness for all contact pairs in a linear perturbation step:

**CONTACT CONTROLS*, PERTURBATION TANGENT SCALE FACTOR=*factor*

To modify the tangential penalty stiffness for a specific contact pair in a linear perturbation step:

**CONTACT CONTROLS*, PERTURBATION TANGENT SCALE FACTOR=*factor*, SECONDARY=*secondary surface*, MAIN=*main surface*

Abaqus/CAE Usage: Modifying the tangential penalty stiffness in linear perturbation steps is not supported in Abaqus/CAE.

Contact Pressure–Dependent Constraint Enforcement in Linear Perturbation Steps

Except for the LCP perturbation procedure, contact constraints are usually fully enforced during perturbation steps for all closed contact interfaces independent of local normal pressure in the base state. You can use two control pressure coefficients, p_0 and p_1, to relax the constraints that have low pressure in the base state or even completely remove them. Both normal and tangential constraints are affected. For pressures less than p_0 in the base state, the normal and tangential constraints are effectively removed by setting the constraint stiffness to zero. For pressures greater than p_1, the constraints are enforced fully. For pressures between p_0 and p_1, the constraint stiffness is reduced and ramps up linearly between p_0 and p_1. In this pressure range finite contact stiffness is in effect even for contact constraints that would otherwise use strict Lagrangian multiplier enforcement. The initial stress stiffness terms are scaled as well.

All other controls for normal and tangential contact penalties are applicable. Constraints open in the base state are unaffected. Pressure-dependent constraint enforcement cannot be used during general steps.

You can specify the contact pressure–dependent constraint enforcement on a step-by-step basis. This specification affects the perturbation step in which it is specified; it will not carry over to subsequent steps. If you want the same specification applied in a series of perturbation steps, you must specify it explicitly in each step.

Some procedures that rely on a frequency analysis, such as complex frequency analysis and subspace-based steady-state dynamic analysis, are influenced by the specification that was in effect for the prior frequency analysis and the specification that is in effect for these steps. In such cases consistent specification is recommended for these steps. For other mode-based procedures based on a frequency analysis, the specification is ignored and only the effect of the previous frequency analysis is considered.

Pressure-dependent constraint enforcement is not available for the static LCP perturbation procedure (*Linear Complementarity Problem (LCP) Solution Technique for Solving Contact Problems*).

Input File Usage: To specify base-state pressure-dependent constraint enforcement for all contact pairs in a linear perturbation step:

**CONTACT CONTROLS*, PRESSURE DEPENDENT PERTURBATION=p_1
p_0

To specify base-state pressure-dependent constraint enforcement for a specific contact pair in a linear perturbation step:

**CONTACT CONTROLS*, PRESSURE DEPENDENT PERTURBATION=p_1, SECONDARY=*secondary surface*, MAIN=*main surface*
p_0

Abaqus/CAE Usage: Base-state pressure-dependent constraint enforcement in linear perturbation steps is not supported in Abaqus/CAE.

Contact Controls Associated with Second-Order Faces

Second-order elements not only provide higher accuracy but also capture stress concentrations more effectively and are better for modeling geometric features than first-order elements. Surfaces based on second-order element types work well with the surface-to-surface contact formulation but, in some cases, do not work well with the node-to-surface formulation (see *Contact Formulations in Abaqus/Standard* for a discussion of these contact formulations).

Some second-order element types are not well-suited for underlying the secondary surface with the combination of a node-to-surface contact formulation and strict enforcement of “hard” contact conditions because of the distribution of equivalent nodal forces when a pressure acts on the face of the element. As shown in *Figure 1*, a constant pressure applied to the face of a second-order element without a midface node produces forces at the corner nodes acting in the opposite sense of the pressure.

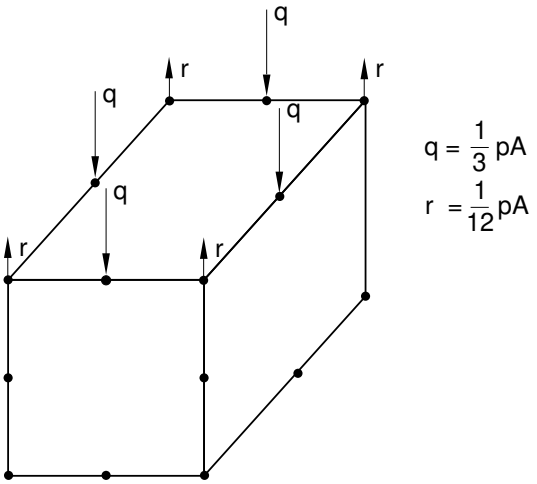


Figure 1: Equivalent nodal loads produced by a constant pressure on the second-order element face in “hard” contact simulations.

This ambiguous nature of the nodal forces in second-order elements can cause Abaqus/Standard to alter its internal contact logic inadequately. Secondary surfaces based on second-order tetrahedral elements can also be problematic for the node-to-surface contact formulation because the distribution of equivalent nodal forces for a pressure acting on a face of these elements is such that the corner nodes have zero force.

Options available in Abaqus/Standard to make it easier to use node-to-surface contact pairs involving second-order secondary faces are discussed below. You can also avoid potential difficulties by using the surface-to-surface contact formulation, which is generally preferable.

Manually or Automatically Adjusting Element Types

Modified 10-node tetrahedral elements (C3D10M, etc.) do not cause fundamental difficulties for the node-to-surface contact formulation and often provide a viable alternative to 10-node second-order tetrahedral elements (C3D10, C3D10HS, etc.) for models with node-to-surface contact pairs. Trade-offs in characteristics of modified 10-node tetrahedral elements versus second-order tetrahedral elements are discussed in *Modified Triangular and Tetrahedral Elements*. If desired, you must make this adjustment to the element type as it does not occur automatically.

Abaqus/Standard automatically adds midface nodes to underlying (serendipity) elements of most 8-node secondary facets associated with non-tied node-to-surface contact pairs. For the three-dimensional 18-node gasket elements, the midface nodes are also generated automatically if they are not given in the element connectivity. The presence of the midface node results in a distribution of nodal forces that is not ambiguous for the contact algorithm. The element families C3D20(RH), C3D15(H), S8R5, and M3D8 are converted to the families C3D27(RH), C3D15V(H), S9R5, and M3D9, respectively. Since Abaqus/Standard does not convert second-order coupled temperature-displacement, coupled thermal-electrical-structural, and coupled pore pressure–displacement elements, you should use an alternative method to avoid problems with serendipity elements in the node-to-surface contact formulation in those cases. Abaqus/Standard will interpolate nodal quantities, such as temperature and field variables, at the automatically generated midface nodes when values are prescribed at any of the user-defined nodes. Abaqus/Standard does not convert second-order serendipity elements if the secondary surface is used in a tied contact pair.

By default, Abaqus/Standard does not automatically add midface nodes to second-order serendipity elements that form a secondary surface for surface-to-surface contact pairs; however, an option is available to enable the same algorithm for automatically adding midface nodes as used by node-to-surface contact pairs.

Input File Usage: **CONTACT PAIR*, TYPE=SURFACE TO SURFACE, MIDFACE NODES=YES

Abaqus/CAE Usage: You cannot enable automatic conversion of serendipity elements underlying secondary surfaces of surface-to-surface contact pairs in Abaqus/CAE.

Supplementary Contact Constraints

Another approach to avoiding difficulties that certain element types present to the node-to-surface contact formulation is to add supplementary contact constraints without changing the underlying element formulation. This approach is applicable only to cases in which node-to-surface contact pairs use penalty or augmented Lagrange constraint enforcement or a softened pressure-overclosure relationship, because it would result in overconstrained conditions if strictly enforced "hard" contact conditions are in effect. Supplementary contact constraints are sometimes helpful for improving convergence behavior or for improving the smoothness and accuracy of the contact pressure and underlying element stress; however, the extra constraints present some risk of degrading convergence behavior. Supplementary constraints are used selectively by default for node-to-surface contact pairs with 6-node secondary faces of non-modified elements and 8-node secondary faces unless strictly enforced "hard" contact conditions are in effect. You can deactivate supplementary constraints or add activate supplementary constraints for additional second-order element types underlying the secondary surface.

Input File Usage: **CONTACT PAIR*, INTERACTION=*interaction_property_name*,
SUPPLEMENTARY CONSTRAINTS=SELECTIVE
secondary_surface_name, *main_surface_name*

Use the following option to add supplementary contact constraints for additional second-order element types:

**CONTACT PAIR*, INTERACTION=*interaction_property_name*,
SUPPLEMENTARY CONSTRAINTS=YES
secondary_surface_name, *main_surface_name*

Use the following option to forgo supplementary contact constraints:

**CONTACT PAIR*, INTERACTION=*interaction_property_name*,
SUPPLEMENTARY CONSTRAINTS=NO
secondary_surface_name, *main_surface_name*

Abaqus/CAE Usage: For a node-to-surface contact formulation:

Interaction module: **Create Interaction**: **Surface-to-surface contact (Standard)**: select the main surface; click **Surface**; select the secondary surface; Interaction editor; **Use supplementary contact points**: **Selectively**, **Always**, or **Never**; **Contact interaction property**: *interaction_property_name*

Smoothness of Contact Force Redistribution upon Sliding for Surface-to-Surface Contact Pairs

You can control the smoothness of nodal contact force redistribution upon sliding for surface-to-surface contact pairs. The default setting, which is generally appropriate, results in the smoothness of the nodal force redistribution being of the same order as the elements underlying the secondary surface; that is, linear redistribution smoothness

for linear elements, and quadratic redistribution smoothness for second-order elements. Quadratic redistribution smoothness usually tends to improve convergence behavior and improve resolution of contact stresses within regions of rapidly varying contact stresses. However, quadratic redistribution smoothness tends to increase the number of nodes involved in each constraint, which can increase the computational cost of the equation solver. Linear redistribution smoothness tends to provide better resolution of contact stresses near edges of active contact regions and, therefore, occasionally results in better convergence behavior.

Input File Usage: Use the following option to indicate that the smoothness of the contact force redistribution upon sliding should be of the same order as the elements underlying the secondary surface for surface-to-surface contact pairs:

**CONTACT PAIR*, TYPE=SURFACE TO SURFACE, SLIDING
TRANSITION=ELEMENT ORDER SMOOTHING
secondary_surface_name, *main_surface_name*

Use the following option to indicate linear smoothness of the contact force redistribution upon sliding for surface-to-surface contact pairs:

**CONTACT PAIR*, TYPE=SURFACE TO SURFACE, SLIDING
TRANSITION=LINEAR SMOOTHING
secondary_surface_name, *main_surface_name*

Use the following option to indicate quadratic smoothness of the contact force redistribution upon sliding for surface-to-surface contact pairs:

**CONTACT PAIR*, TYPE=SURFACE TO SURFACE, SLIDING
TRANSITION=QUADRATIC SMOOTHING
secondary_surface_name, *main_surface_name*

Abaqus/CAE Usage: You cannot change the default contact force redistribution in Abaqus/CAE.

Contact Controls Specific to General Contact in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About General Contact in Abaqus/Standard*
- **CONTACT*
- **CONTACT FORMULATION*
- **CONTACT CONTROLS*
- *Specifying contact formulation assignments for general contact*

Overview

Numerical controls associated with the general contact algorithm in Abaqus/Standard:

- should not be modified from their default settings for the majority of problems;
- can be used for problems where the default settings do not provide cost-effective solutions;
- can be used to control the main-secondary roles and the sliding formulation; and
- in some cases can be applied selectively to particular regions within a general contact domain.

Contact Formulation

By default, the general contact algorithm uses the finite-sliding, surface-to-surface contact formulation, which is discussed in *Contact Formulations in Abaqus/Standard*. In addition, it is supplemented by the edge-to-surface, edge-to-edge and vertex-to-surface formulations, which are also based on the finite-sliding tracking approach. Optionally, you can specify the small-sliding tracking approach for portions of the general contact domain and, by extension, the entire general contact domain.

Constraint Enforcement Method

The general contact algorithm uses a penalty method to enforce active contact constraints by default. Other constraint enforcement methods can be specified as part of the surface interaction (i.e., contact property) definition, as discussed in *Contact Constraint Enforcement Methods in Abaqus/Standard*. Assignment of contact properties to general contact interactions is discussed in *Contact Properties for General Contact in Abaqus/Standard*.

Numerical Controls for Friction

Numerical controls associated with friction are discussed in *Frictional Behavior*.

Beam-to-Beam Contact

Activation of beam-to-beam contact is discussed in *About General Contact in Abaqus/Standard*.

Main and Secondary Surface Roles of a Contact Formulation

The surface-to-surface contact formulation used by general contact generates individual contact constraints using a main-secondary approach, as discussed in *Contact Formulations in Abaqus/Standard*. Abaqus/Standard assigns default pure main-secondary surface roles of a contact formulation for contact involving disconnected bodies within the general contact domain. Bodies consisting of connected beam and truss elements are considered disconnected bodies even though these bodies may share nodes with other faceted bodies. Internal surfaces are generated automatically using the naming convention `General_Contact_Faces_k`, where `k` corresponds to an automatically assigned component number. By default, the lower-numbered component surfaces act as main surfaces to the higher-numbered component surfaces. An exception is when component surfaces consisting of beam and truss elements interact with faceted component surfaces in the edge-to-surface contact formulation. A component surface consisting of beam and truss elements acts as the main surface in the edge-to-surface formulation if half of the average element radius is larger than the average smallest facet length of the faceted component surface. You can determine the default pure main-secondary roles by viewing the automatically generated internal surfaces in the Visualization module of Abaqus/CAE (see *Using display groups to display subsets of your model*). Self-contact within a body is treated with balanced main-secondary contact by default, with each surface node acting as a main node in some constraints and as a secondary node in other constraints.

For example, if the general contact domain spans three disconnected bodies, the following three internal "component-surfaces" for general contact are created automatically:

- `General_Contact_Faces_1`
- `General_Contact_Faces_2`
- `General_Contact_Faces_3`

By default, the first surface listed acts as a main surface to the other two, and `General_Contact_Faces_2` acts as a main surface to `General_Contact_Faces_3`. If any of these surfaces contain beam or truss elements interacting with other faceted surfaces in the edge-to-surface contact formulation, the decision to use these as the main surfaces will depend on the average element radius and the average smallest facet length of the faceted surfaces. By default, self-contact within each of these three surfaces is modeled with balanced main-secondary contact.

All XFEM-based crack surfaces in the general contact domain are assigned to a separate component and assigned the highest component number. Therefore, crack surfaces act as secondary surfaces s to other components by default. Contact between portions of crack surfaces are handled with balanced main-secondary contact since they all belong to a single component.

Specifying Nondefault Main-Secondary Roles

You can override the default main-secondary roles by specifying pure main-secondary roles or by specifying that balanced main-secondary contact should be used. The default main-secondary treatment works well in most cases. Keep the following points in mind when modifying the main-secondary assignments, in addition to other factors discussed in this section:

- Do not use the internally generated component surfaces when assigning alternative main-secondary roles (instead, use surface names that you define). Named XFEM-based crack surfaces are not supported for specifying main-secondary roles.
- The main-secondary role assignments are part of the model definition and cannot be modified from step to step.
- The guidelines for assigning pure main-secondary roles for contact pairs discussed in *Defining Contact between Two Separate Surfaces* are also applicable for situations in which you reassign pure main-secondary roles for general contact.

- The limitations of balanced (symmetric) main-secondary contact pairs discussed in *Using Symmetric Main-Secondary Contact Pairs to Improve Contact Modeling* are also applicable for situations in which you reassign balanced main-secondary contact for general contact. Balanced main-secondary contact can result in reduced robustness due to the increased number of constraints and the possibility of overconstraints.

Input File Usage: Use the following option to indicate that the first surface should be considered the secondary surface:

**CONTACT FORMULATION*, TYPE=MAIN SECONDARY ROLES
surf_1, *surf_2*, SECONDARY

Use the following option to indicate that the first surface should be considered the main surface:

**CONTACT FORMULATION*, TYPE=MAIN SECONDARY ROLES
surf_1, *surf_2*, MAIN

If the first surface name is omitted, a default surface that encompasses the entire general contact domain is assumed. The second surface name must be specified.

Use the following option to specify that contact with balanced main and secondary surface roles should be used between two surfaces:

**CONTACT FORMULATION*, TYPE=MAIN SECONDARY ROLES
surf_1, *surf_2*, BALANCED

If the first surface name is omitted, a default surface that encompasses the entire general contact domain is assumed. If the second surface name is omitted, contact between the first surface and itself is assumed.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Contact Formulation**: **Main-secondary assignments: Edit**: select the surfaces in the columns on the left, and click the arrows in the middle to transfer them to the list of main-secondary assignments.

In the **First Surface Type** column, enter SECONDARY to indicate that the first surface should be considered the secondary surface, enter MAIN to indicate that the first surface should be considered the main surface, or enter BALANCED to specify that balanced main-secondary contact should be used between the two surfaces.

Automatically Generated Contact Exclusions

Abaqus/Standard automatically generates contact exclusions for the main-secondary roles opposite to specified pure main-secondary roles; therefore, self-contact is excluded for any regions of the two surfaces that overlap. For example, specifying that the general contact interaction between *surf_A* and *surf_B* should use pure main-secondary contact with *surf_A* considered to be the secondary surface would result in exclusions being generated internally for main faces of *surf_A* contacting secondary faces of *surf_B*; self-contact would be excluded for the region of overlap between *surf_A* and *surf_B*. An error message is issued if the second surface name is omitted or is the same as the first surface name since this input would result in the exclusion of self-contact for the surface.

Specifying Small Sliding within General Contact

You can specify the small-sliding tracking approach for interactions involving portions of the entire general contact domain and, by extension, the entire contact domain. The small-sliding approach avoids repeated contact tracking and the need to re-establish the nodes involved in the constraint connectivity, which makes it more efficient than the default finite-sliding approach for general contact. However, you should be aware of the approximations involved in small sliding to decide whether they are appropriate (*The Small-Sliding Tracking Approach*). The small-sliding approach involves only the surface-to-surface contact over opposing surfaces; edges and vertices are deactivated. The surface pairs you specify to utilize small sliding are excluded automatically for the default finite-sliding approach. Subsequently, none of the types of contact based on finite-sliding assumptions (such as surface-to-surface contact or edge and vertex contact) are active over the pairings with the small-sliding specification.

You should not use the internally generated component surfaces when assigning the small-sliding approach; instead, you should define the surface names. Named XFEM-based crack surfaces are not allowed for specifying the small-sliding approach.

Input File Usage: Use the following option to specify the surface pairings to consider for the small-sliding approach:

**CONTACT FORMULATION*, TYPE=SLIDING FORMULATION
surf_1, *surf_2*, SMALL SLIDING

Use the following option to specify the small-sliding approach for all interactions involving *surf_1*:

**CONTACT FORMULATION*, TYPE=SLIDING FORMULATION
, *surf_1*, SMALL SLIDING

Use the following option to specify the small-sliding approach everywhere (first data line) except contact involving *surf_1* with *surf_2* (where the default finite-sliding approach is active):

**CONTACT FORMULATION*, TYPE=SLIDING FORMULATION
, , SMALL SLIDING
surf_1, *surf_2*,

A blank entry in the first and second data entries of the data lines refers to the default surface that encompasses the entire general contact domain.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Contact Formulation**: **Sliding Formulation assignments: Edit**: select the surfaces in the columns on the left, and click the arrows in the middle to transfer them to the list of sliding formulation assignments.

In the **Sliding** column, select Small Sliding, or None to indicate the sliding formulation.

Smoothness of Contact Force Redistribution upon Sliding

You can control the smoothness of nodal contact force redistribution upon sliding. The default setting, which is generally appropriate, results in the smoothness of the nodal force redistribution being of the same order as the elements underlying the secondary surface; that is, linear redistribution smoothness for linear elements, and

quadratic redistribution smoothness for second-order elements. Quadratic redistribution smoothness usually tends to improve convergence behavior and improve resolution of contact stresses within regions of rapidly varying contact stresses. However, quadratic redistribution smoothness tends to increase the number of nodes involved in each constraint, which can increase the computational cost of the equation solver. Linear redistribution smoothness tends to provide better resolution of contact stresses near edges of active contact regions and, therefore, occasionally results in better convergence behavior.

Input File Usage: Use the following option to indicate that the smoothness of the contact force redistribution upon sliding should be of the same order as the elements underlying the secondary surface:

**CONTACT FORMULATION*, TYPE=SLIDING TRANSITION
surf_1, *surf_2*, ELEMENT ORDER SMOOTHING

If the first surface name is omitted, a default surface that encompasses the entire general contact domain is assumed. If the second surface name is omitted, contact between the first surface and itself is assumed.

Use the following option to indicate linear smoothness of the contact force redistribution upon sliding:

**CONTACT FORMULATION*, TYPE=SLIDING TRANSITION
surf_1, *surf_2*, LINEAR SMOOTHING

If the first surface name is omitted, a default surface that encompasses the entire general contact domain is assumed. If the second surface name is omitted, contact between the first surface and itself is assumed.

Use the following option to indicate quadratic smoothness of the contact force redistribution upon sliding:

**CONTACT FORMULATION*, TYPE=SLIDING TRANSITION
surf_1, *surf_2*, QUADRATIC SMOOTHING

If the first surface name is omitted, a default surface that encompasses the entire general contact domain is assumed. If the second surface name is omitted, contact between the first surface and itself is assumed.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Contact Formulation**: **Sliding Transition assignments: Edit**: select the surfaces in the columns on the left, and click the arrows in the middle to transfer them to the list of sliding transition assignments.

In the **Smoothness** column, enter QUADRATIC_SMOOTHING, LINEAR_SMOOTHING, or ELEMENT_ORDER_SMOOTHING to indicate the smoothness of the contact force redistribution upon sliding.

Additional Global Numerical Controls for General Contact

Some additional numerical contact controls can be modified globally from step-to-step for general contact; you cannot specify contact controls for individual surface pairings within the general contact domain. You can apply contact stabilization to address rigid body modes that occur prior to the establishment of contact in the model, and you can adjust the tolerances used by Abaqus/Standard to determine contact penetrations and separations; both techniques are discussed in *Generally Applicable Contact Controls in Abaqus/Standard*.

Contact Formulations, Numerical Methods, and Controls in Abaqus/Explicit

In this section:

- *Contact Formulation for General Contact in Abaqus/Explicit*
- *Contact Formulations for Contact Pairs in Abaqus/Explicit*
- *Contact Constraint Enforcement Methods in Abaqus/Explicit*
- *Contact Controls for General Contact in Abaqus/Explicit*
- *Contact Controls for Contact Pairs in Abaqus/Explicit*

Contact Formulation for General Contact in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *About General Contact in Abaqus/Explicit*
- **CONTACT*
- **CONTACT FORMULATION*
- *Specifying contact formulation assignments for general contact*

Overview

The contact formulation used with the general contact algorithm in Abaqus/Explicit:

- includes the contact surface weighting, surface polarity, and the sliding formulation; and
- can be applied selectively to particular regions within a general contact domain.

The general contact formulation uses a penalty method to enforce contact constraints between surfaces; the constraint enforcement method is discussed in *Contact Constraint Enforcement Methods in Abaqus/Explicit*.

Specifying the Contact Formulation

Currently you can specify only the contact surface weighting and polarity for the general contact algorithm. The contact formulation propagates through all analysis steps in which the general contact interaction is active.

The surface names used to specify the regions where a nondefault contact formulation should be assigned do not have to correspond to the surface names used to specify the general contact domain. In many cases the contact interaction will be defined for a large domain, while a nondefault contact formulation will be assigned to a subset of this domain. Any contact formulation assignments for regions that fall outside the general contact domain will be ignored. The last assignment will take precedence if the specified regions overlap.

Input File Usage: **CONTACT FORMULATION*

This option must be used in conjunction with the **CONTACT* option. It should appear at most once per step for each value of the TYPE parameter; the data line can be repeated as often as necessary to assign contact formulations to different regions.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Contact Formulation**

Contact Surface Weighting

Generally, contact constraints in a finite element model are applied in a discrete manner, meaning that for hard contact a node on one surface is constrained to not penetrate the other surface. In pure main-secondary contact the node with the constraint is part of the secondary surface and the surface with which it interacts is called the main surface. For balanced main-secondary contact Abaqus/Explicit calculates the contact constraints twice for each set of surfaces in contact, in the form of penalty forces: once with the first surface acting as the main surface

and once with the second surface acting as the main surface. The weighted average of the two corrections (or forces) is applied to the contact interaction.

Balanced main-secondary contact minimizes the penetration of the contacting bodies and, thus, provides better enforcement of contact constraints and more accurate results in most cases. In pure main-secondary contact the nodes on the main surface can, in principle, penetrate the secondary surface unhindered (see *Figure 1*).

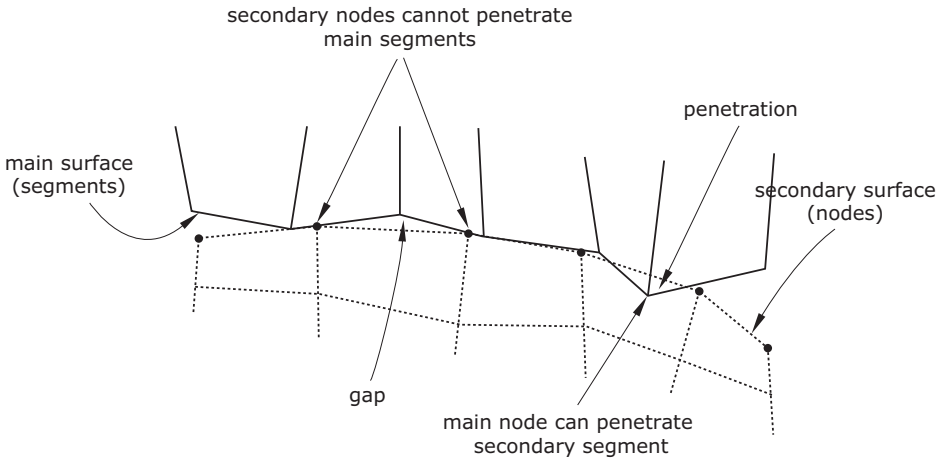


Figure 1: Main surface penetrations into the secondary surface in pure main-secondary contact due to coarse discretization.

The general contact algorithm in Abaqus/Explicit uses balanced main-secondary weighting whenever possible; pure main-secondary weighting is used for contact interactions involving node-based surfaces, which can act only as pure secondary surfaces and for contact interactions involving analytical rigid surfaces, which can act only as pure main surfaces. Surface-based cohesive behavior and no-separation behavior (without rough friction) use a pure main-secondary algorithm by default (see *Considerations for Contact Cohesive Behavior and No-Separation Behavior*). You can choose to specify a pure main-secondary weighting for other interactions as well.

There is no main-secondary relationship for edge-to-edge contact; both contacting edges are given equal weighting.

Specifying Pure Main-Secondary Weighting for Node-to-Face Contact

You can specify that a general contact interaction should use pure main-secondary weighting for node-to-face contact. This specification has no effect on edge-to-edge contact and cannot be used to make a node-based surface act as a main surface. When two originally flat surfaces contact one another, a more uniform penetration distance distribution (and consequently pressure distribution) might result with pure main-secondary weighting where the more refined surface acts as the secondary surface as compared to balanced main-secondary weighting. This can be particularly evident if the mesh densities of the contacting surfaces differ significantly—with balanced weighting the contact penetrations will be smaller near the nodes of the coarsely meshed surface.

Abaqus/Explicit automatically generates contact exclusions for the main-secondary orientation opposite to that specified. For example, specifying that the general contact interaction between *surf_A* and *surf_B* should use pure main-secondary weighting with *surf_A* considered to be the secondary surface results in exclusions being generated internally for faces of *surf_A* contacting nodes of *surf_B*. However, if *surf_A* and *surf_B* overlap such that node-to-face self-contact exists, the balanced main-secondary weighting is used within the overlap region of *surf_A* and *surf_B*. Abaqus/Explicit issues a warning message if the second surface name is omitted or is the same as the first surface name because this input would result in node-to-face self-contact for the surface. If the secondary surface (*surf_A*) continuously resides within a larger surface that is defined within the contact domain, it is possible for nodes of the main surface (*surf_B*) to contact faces that are adjacent to *surf_A*. Therefore, the "perimeter" region of *surf_A* would tend toward a balanced main-secondary weighting (rather than a pure main-secondary weighting).

Input File Usage: Use the following option to indicate that the first surface should be considered the secondary surface (default):

**CONTACT FORMULATION*, TYPE=PURE MAIN-SECONDARY
surf_1, *surf_2*, SECONDARY

Use the following option to indicate that the first surface should be considered the main surface:

**CONTACT FORMULATION*, TYPE=PURE MAIN-SECONDARY
surf_1, *surf_2*, MAIN

If the first surface name is omitted, a default surface that encompasses the entire general contact domain is assumed. The second surface name must be specified.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Contact Formulation**: **Pure main-secondary assignments: Edit**: select the surfaces in the columns on the left, and click the arrows in the middle to transfer them to the list of main-secondary assignments.

In the **First Surface Type** column, enter SECONDARY to indicate that the first surface should be considered the secondary surface, and enter MAIN to indicate that the first surface should be considered the main surface.

Considerations for Contact Cohesive Behavior and No-Separation Behavior

Pure main-secondary roles are assigned for contact cohesive behavior and no-separation behavior by default to avoid locking mechanisms that often occur for bonded interfaces under bending. The first surface specified in the assignment of a cohesive contact property assignment acts as the secondary surface by default. For no-separation behavior, the secondary surface is assigned to be the more refined surface; if both surfaces are equally refined, the first surface specified is assigned to be the secondary surface.

You can override the default by directly specifying which surface should act as the secondary surface or that balanced main-secondary weighting should be used.

Input File Usage: Use the following option to indicate that a balanced main-secondary formulation should be considered (default):

**CONTACT FORMULATION*, TYPE=MAIN SECONDARY ROLES
surf_1, *surf_2*, BALANCED

Use the following option to indicate that the first surface should be considered the secondary surface:

**CONTACT FORMULATION*, TYPE=MAIN SECONDARY ROLES
surf_1, *surf_2*, SECONDARY

Use the following option to indicate that the first surface should be considered the main surface:

**CONTACT FORMULATION*, TYPE=MAIN SECONDARY ROLES
surf_1, *surf_2*, MAIN

If the first surface name is omitted, a default surface that encompasses the entire general contact domain is assumed. In this case you must specify the second surface name.

Abaqus/CAE Usage: The alternative interface to control main-secondary roles is not supported in Abaqus/CAE.

Contact Surface Polarity

By default, general contact considers both sides of all double-sided elements in surfaces specified to be included for contact purposes (side labels of double-sided elements are ignored). This default can be overridden for node-to-face and Eulerian-Lagrangian contact and in some cases results in more accurate enforcement of contact.

Surface polarity is not considered for edge-to-edge contact, including edges activated on faces of solid elements.

Specifying Surface Polarity for Node-to-Face and Eulerian-Lagrangian Contact

Changing the polarity of double-sided elements forces the contact algorithm to treat them as if they were solid elements. More accuracy might be gained by converting double-sided elements to single-sided if there is a chance that secondary nodes might be "caught" behind the surface in node-to-face contact or if material contained on one side of a double-sided surface leaks to the other side in Eulerian-Lagrangian contact. Improvements in performance and memory use might also be observed with Eulerian-Lagrangian contact if double-sided Lagrangian surfaces are converted to single-sided for contact with all Eulerian material surfaces.

Input File Usage: Use the following option to indicate that the sides of the (double-sided) elements specified in the second surface's definition should be considered for contact with the first surface:

**CONTACT FORMULATION*, TYPE=POLARITY
surf_1, *surf_2*

Use the following option to indicate that the SPOS side of the (double-sided) elements in the second surface should be considered for contact with the first surface:

**CONTACT FORMULATION*, TYPE=POLARITY
surf_1, *surf_2*, SPOS

Use the following option to indicate that the SNEG side of the (double-sided) elements in the second surface should be considered for contact with the first surface:

**CONTACT FORMULATION*, TYPE=POLARITY
surf_1, *surf_2*, SNEG

Use the following option to indicate that both sides of the (double-sided) elements in the second surface should be considered for contact with the first surface:

**CONTACT FORMULATION*, TYPE=POLARITY
surf_1, *surf_2*, TWO SIDED

If the first surface name is omitted, a default surface that encompasses the entire general contact domain is assumed. The second surface name must be specified.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Contact Formulation**: **Polarity assignments: Edit**: select the surfaces in the columns on

the left, and click the arrows in the middle to transfer them to the list of polarity assignments.

In the **Second Surface Type** column, enter SPOS to indicate that the SPOS side of the (double-sided) elements in the second surface should be considered for contact with the first surface, enter SNEG to indicate that the SNEG side of the (double-sided) elements in the second surface should be considered for contact with the first surface, and enter TWO_SIDED to indicate that both sides of the (double-sided) elements in the second surface should be considered for contact with the first surface.

Sliding Formulation

Currently only the finite-sliding formulation is available for general contact in Abaqus/Explicit. This formulation allows for arbitrary separation, sliding, and rotation of the surfaces in contact. For cases in which small-sliding or infinitesimal-sliding assumptions would be preferred, the contact pair algorithm should be used (see *Contact Formulations for Contact Pairs in Abaqus/Explicit*).

Abaqus/Explicit is designed to simulate highly nonlinear events or processes. Because it is possible for a node on one surface to contact any of the facets on the opposite surface, Abaqus/Explicit must use sophisticated search algorithms for tracking the motions of the surfaces. The finite-sliding contact search algorithm is designed to be robust, yet computationally efficient. This algorithm assumes that the incremental relative tangential motion between surfaces does not significantly exceed the dimensions of the main surface facets, but there is no limit to the overall relative motion between surfaces. It is rare for the incremental motion to exceed the facet size because of the small time increment used in explicit dynamic analyses. In cases involving relative surface velocities that exceed material wave speeds it might be necessary to reduce the time increment.

The contact search algorithm uses a global search when a contact interaction is first introduced, and a hierarchical global/local search algorithm is used thereafter. No user control of the search algorithm is needed.

Local Tangent Directions for Contact

Local tangent directions for contact provide a reference frame for select general contact output variables in Abaqus/Explicit (see *About General Contact in Abaqus/Explicit*). These local tangent directions are separate from local coordinate systems associated with user subroutines *VFRICTION* and *VUINTERACTION*.

Abaqus/Explicit establishes and updates the orientation of the first local contact tangent direction, $\mathbf{t}_1$, at secondary nodes and edge nodes according to the logic described below for different contact formulation types within general contact. The orientation of the second local tangent direction, $\mathbf{t}_2$, is found as the cross product of the contact normal direction, $\mathbf{n}$ with $\mathbf{t}_1$. A change in the predominant contact formulation type that is active at a node might lead to a sudden change in the local tangent directions.

Finite-sliding, node-to-surface formulation for nonanalytical surfaces

The $\mathbf{t}_1$-direction is initialized at a secondary node on first contact using the standard convention for calculating a first local surface tangent direction (see *Conventions*) or using the user orientation specified through the option **SURFACE PROPERTY ASSIGNMENT*, PROPERTY=ORIENTATION (applicable only for nodes belonging to an element-based surface). In subsequent increments, if the secondary node belongs to an element-based surface, the $\mathbf{t}_1$-direction rotates with the secondary surface for geometrically nonlinear analyses; otherwise, the standard convention is used.

Finite-sliding, node-to-surface formulation with an analytical surface

The $\mathbf{t}_1$-direction for contact is initialized at a secondary node on first contact to be aligned with the convention for the $\mathbf{t}_1$-direction of the analytical rigid surface discussed in *Analytical Rigid Surface Definition* at the point of contact. In subsequent increments, the $\mathbf{t}_1$-direction for contact at a secondary node evolves such that it continues to be aligned with the $\mathbf{t}_1$-direction of the analytical rigid surface at the current point of contact.

Finite-sliding, edge-to-edge formulation

The $\mathbf{t}_1$-direction for an edge-to-edge contact constraint is initialized on first contact to be in the axial direction of one of the edges involved in the contact and will evolve to remain aligned with the axial direction of this edge until a local transition to another edge occurs, and then the axial direction of that edge will be adopted as the $\mathbf{t}_1$-direction.

Local surface tangent directions will often differ across a contact interface. For example, respective local $\mathbf{t}_1$-directions (CTANDIR1) on opposite sides of an interface will evolve differently if surface rotations across the interface are not the same. The respective local $\mathbf{t}_2$-directions (CTANDIR2) on opposite sides of an interface are typically in opposing directions initially, due to secondary nodes on opposite sides of an interface having opposing contact normal directions.

Contact Formulations for Contact Pairs in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *About Surfaces*
- *About Contact Pairs in Abaqus/Explicit*
- **CONTACT PAIR*
- *Defining surface-to-surface contact*

Overview

The contact formulation for the contact pair algorithm in Abaqus/Explicit includes:

- the contact surface weighting (balanced or pure main-secondary); and
- the sliding formulation (finite, small, or infinitesimal).

You can also specify the method that is used to enforce contact constraints in the contact pair; these methods are discussed in *Contact Constraint Enforcement Methods in Abaqus/Explicit*.

Contact Surface Weighting

Both the pure main-secondary and the balanced main-secondary contact algorithms are available in Abaqus/Explicit. By default, Abaqus/Explicit will decide which algorithm to use for any given contact pair based on the nature of the two surfaces forming the contact pair and whether kinematic or penalty enforcement of contact constraints is used. You can override the defaults in some cases.

Default Choices for the Contact Pair Weighting

Abaqus/Explicit uses the pure main-secondary, kinematic contact algorithm, by default, in the following situations (the first surface in each situation listed is designated the main surface):

- when a rigid surface contacts a deformable surface;
- when an element-based surface contacts a node-based surface; or
- when a surface based on continuum elements contacts a surface based on shell or membrane elements.

By default, Abaqus/Explicit uses the balanced main-secondary, kinematic contact algorithm in the following situations:

- when a single surface contacts itself (referred to as self-contact or single-surface contact); or
- when two deformable surfaces that are meshed with similar elements (i.e., either both surfaces have shells or membranes or both have continuum elements) contact each other.

If the penalty contact algorithm is specified, Abaqus/Explicit uses pure main-secondary weighting, by default, in the following situations (the first surface in each situation listed is designated the main surface):

- when an analytical rigid surface contacts a deformable surface; or
- when an analytical rigid surface or an element-based surface contacts a node-based surface.

If the penalty contact algorithm is specified, Abaqus/Explicit chooses balanced main-secondary weighting, by default, in the following situations:

- when a single surface contacts itself (referred to as self-contact or single-surface contact); or
- when two element-based surfaces contact each other.

Balanced main-secondary weighting means that the corrections produced by both sets of contact calculations are weighted equally.

Modifying the Default Choices for the Contact Pair Weighting

When the kinematic contact method is chosen, you can override the default contact pair weighting only when two separate deformable element-based surfaces are contacting each other, which corresponds to the last situation in each list for kinematic contact given in the previous section.

The following aspects should be considered when deciding whether or not to override the default choice. First, the balanced main-secondary contact algorithm requires more computational time, but it is typically more accurate. Second, when the densities differ by orders of magnitude, the less dense body should be a pure secondary surface. Contact-induced noise can occur if a surface on a much denser body is at all weighted as a secondary surface. Finally, to avoid significant penetration for hard contact, the surface with the finer mesh should not be the main surface in the pure main-secondary contact pair.

When the penalty contact method is chosen, you can choose to specify a pure main-secondary weighting to reduce computational time. When two originally flat surfaces contact one another, a more uniform penetration distance distribution (and consequently pressure distribution) may result with pure main-secondary weighting as compared to balanced main-secondary weighting. This can be particularly evident if the mesh densities of the contacting surfaces differ significantly—with balanced weighting the contact penetrations will be smaller near the nodes of the coarsely meshed surface. However, balanced main-secondary weighting provides better enforcement of contact constraints in most cases.

You define a weighting factor, f, to specify the main-secondary weighting. Set f=1.0 to designate the first surface in the contact pair as the main surface and the second surface as the secondary surface. Set f=0.0 to designate the first surface in the contact pair as the secondary surface and the second surface as the main surface. Specifying any value of f between 0 and 1.0 invokes the balanced main-secondary contact algorithm. When f=0.5, which is the default for balanced main-secondary contact pairs, Abaqus/Explicit weights each set of corrections equally. In contrast, Abaqus/Standard uses a pure main-secondary contact algorithm; the secondary surface must always be given first, as in the f=0.0 case above.

Input File Usage: **CONTACT PAIR*, WEIGHT=*f*

Abaqus/CAE Usage: Interaction module: interaction editor: **Weighting factor Specify** *f*

Sliding Formulation

In Abaqus/Explicit there are three approaches to account for the relative motion of the two surfaces forming a contact pair:

- finite sliding, which is the most general and allows any arbitrary motion of the surfaces;
- small sliding, which assumes that although two bodies may undergo large motions, there will be relatively little sliding of one surface along the other; or
- infinitesimal sliding and rotation, which assumes that both the relative motion of the surfaces and the absolute motion of the contacting bodies are small.

The small-sliding and infinitesimal-sliding formulations cannot be used for contact pairs using the penalty contact algorithm or involving self-contact or analytical rigid surfaces.

Using the Finite-Sliding Formulation

The finite-sliding formulation allows for arbitrary separation, sliding, and rotation of the surfaces. Abaqus/Explicit uses this formulation by default. Only the finite-sliding approach is available for self-contact or contact involving analytical rigid surfaces.

Input File Usage: **CONTACT PAIR*

Abaqus/CAE Usage: Interaction module: interaction editor: **Sliding formulation: Finite sliding**

Example

The following input defines finite-sliding contact between the surfaces `ASURF` and `BSURF`, shown in *Figure 1*, with `ASURF` acting as the secondary surface:

```
*SURFACE,NAME=ASURF
ESETA,
*SURFACE,NAME=BSURF
ESETB,
*CONTACT PAIR,INTERACTION=PAIR1, WEIGHT=0.0
ASURF, BSURF
*SURFACE INTERACTION,NAME=PAIR1
```

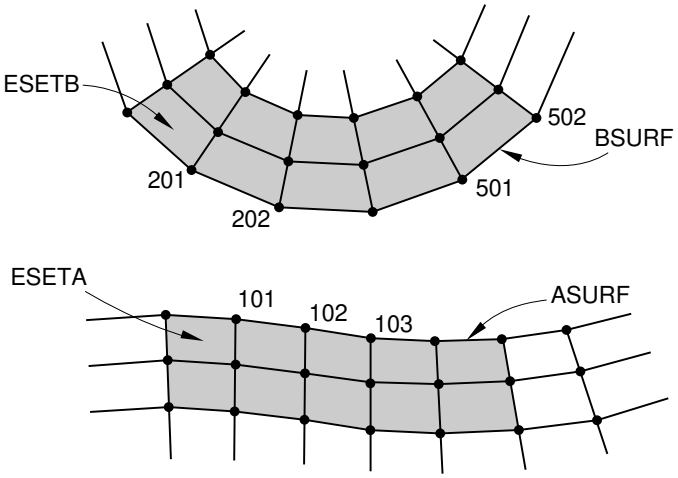


Figure 1: Contacting bodies.

In the example shown in *Figure 1* secondary node 101 may come into contact anywhere along the main surface `BSURF`. While in contact, it is constrained to slide along `BSURF`, irrespective of the orientation and deformation of this surface. This behavior is possible because Abaqus/Explicit tracks the position of node 101 relative to the main surface `BSURF` as the bodies deform. *Figure 2* shows the possible evolution of the contact between node 101 and its main surface `BSURF`. Node 101 is in contact with the element face with end nodes 201 and 202 at time t_1. The load transfer at this time occurs between node 101 and nodes 201 and 202 only. Later on, at time t_2, node 101 may find itself in contact with the element face with end nodes 501 and 502. Then the load transfer will occur between node 101 and nodes 501 and 502.

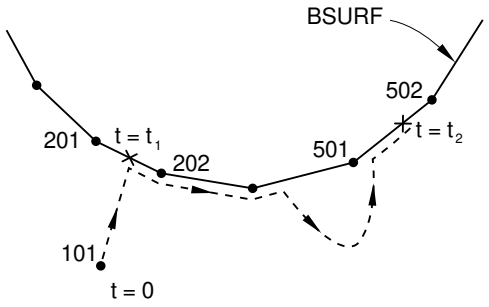


Figure 2: Trajectory of node 101 in finite-sliding contact.

Finite Sliding in a Geometrically Linear Analysis

Finite-sliding simulations usually include nonlinear geometric effects because such simulations generally involve large deformations and large rotations. However, it is also possible to use the finite-sliding formulation in a geometrically linear analysis (see *Geometric Nonlinearity*). The load transfer paths between the surfaces and the contact direction are updated in finite-sliding, geometrically linear analysis. This capability is useful for analyzing finite sliding between two stiff bodies that do not undergo large rotations.

Using the Small-Sliding Formulation

For a large class of contact problems the general tracking of the finite-sliding formulation is unnecessary, even though geometric nonlinearity must be considered. Abaqus/Explicit provides a *small-sliding* contact formulation for such problems. This formulation assumes that the surfaces may undergo arbitrarily large rotations but that a secondary node will interact with the same local area of the main surface throughout the analysis. Contact pairs that use the small-sliding formulation must be defined in the first step of the simulation, although they may remain active after the first step.

A large-displacement formulation (the default) should be used for the step in which the small-sliding contact formulation should be used.

In a small-sliding analysis every secondary node interacts with its own local tangent plane on the main surface (see *Figure 3*). The secondary node is constrained not to penetrate this local tangent plane. Each local tangent plane, which is a line in two dimensions, is defined by an anchor point, $\mathbf{X_0}$, on the main surface and an orientation vector at the anchor point (see *Figure 3*).

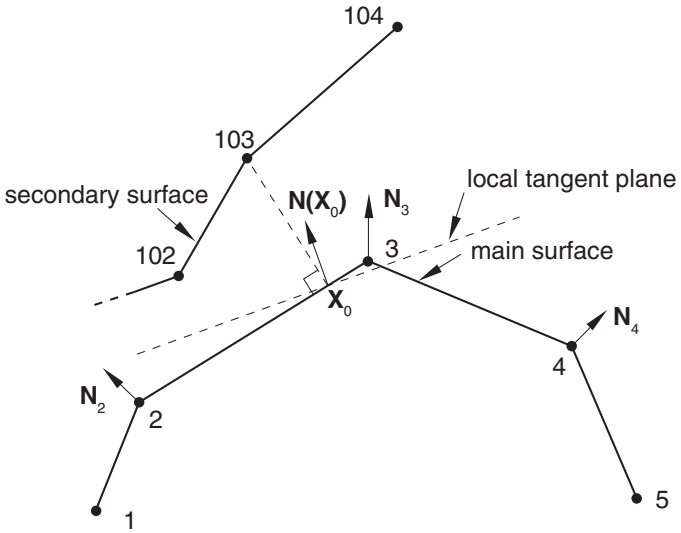


Figure 3: Definition of the anchor point and local tangent plane for node 103.

Having a local tangent plane for each secondary node means that for the small-sliding formulation Abaqus/Explicit does not have to monitor secondary nodes for possible contact along the entire main surface. Therefore, small-sliding contact is less expensive computationally than finite-sliding contact. The cost savings are most dramatic in three-dimensional contact problems.

When the balanced main-secondary contact algorithm is invoked with the small-sliding formulation, anchor points and tangent planes will be computed for both surfaces.

Input File Usage: Use both of the following options:

```
*STEP, NLGEOM=YES
…
*CONTACT PAIR, SMALL SLIDING
```

For example, the following options define small-sliding contact between the two bodies shown in *Figure 1*:

```
*STEP, NLGEOM=YES
…
*SURFACE, NAME=ASURF
ESETA,
*SURFACE, NAME=BSURF
ESETB,
*CONTACT PAIR, SMALL SLIDING, WEIGHT=0.0
ASURF, BSURF
```

Abaqus/CAE Usage: Interaction module: interaction editor: **Sliding formulation: Small sliding**
Step module: step editor: **Nlgeom: On**

Anchor Point and Tangent Plane Definition

The anchor point and the tangent plane orientation are chosen before the analysis starts using the initial configuration of the model. The anchor point and the tangent plane orientation remain fixed with respect to the main surface facet for all steps in which the contact pair is active. No contact constraints are enforced for secondary nodes whose nearest point lies on the free perimeter of the main surface in the original configuration and that do not project onto any main surface facet.

Abaqus/Explicit chooses the anchor point as the nearest point on the main surface. The orientation of the tangent plane is calculated by default from the normals at the main surface nodes, or you can specify it directly.

- Main surface normals: The first step in defining the tangent plane orientation is to construct the unit normal vectors at each node of the main surface. Abaqus/Explicit forms these nodal normals by averaging the normals of the element faces making up the main surface; only the element faces in the surface definition will contribute to the nodal normals. The tangent plane orientation is then calculated from the main surface nodal normals and the element shape functions at the anchor point.

  Figure 3 shows the nodal unit normals for a main surface, the anchor point $\mathbf{X_0}$, and the local tangent plane associated with secondary node 103. Abaqus/Explicit uses the closest point on the main surface as the anchor point. $\mathbf{N}(\mathbf{X_0})$ is the contact direction for secondary node 103 and defines the orientation of the local tangent plane. In this example, as in many cases, the local tangent plane is only an approximation of the actual mesh geometry.
- Main surface normals at symmetry planes: Sometimes the main surface normal and the local tangent plane that Abaqus/Explicit calculates are not suitable for the desired analysis. The most common situation where unsuitable surface normals are calculated occurs when a curved main surface ends at a symmetry plane and

the boundary conditions have been specified in direct format rather than in symmetry "type" format (XSYMM, YSYMM, or ZSYMM—see *Boundary Conditions*). In this case the correct normals should be in the symmetry plane; however, because the surface facets that abut the symmetry plane usually form an angle with the plane, the normal will project away from the symmetry plane. The effect of this behavior can be that a secondary node does not project onto any main surface facet (the secondary node is said not to "intersect" the main surface). No contact constraints will be enforced for such secondary nodes. However, if symmetry "type" format boundary conditions are specified, contact constraints will be enforced as described below. The finite-sliding formulations use no special treatment for main surfaces ending at a symmetry plane.

Figure 4 shows two concentric cylinders that contact each other; the inner cylinder is chosen as the main surface `CSURF`, and a half-symmetry model is used. Since Abaqus/Explicit calculates the nodal normals from the approximate, finite element model, the nodal normal $\mathbf{N_1}$ does not point along the symmetry plane, which means that secondary node 100 has no anchor point within the perimeter of the main surface. Whether or not contact is enforced for node 100 depends on how the symmetry boundary condition is specified. If the individual components are specified rather than a symmetry "type" boundary condition, secondary node 100 will be free to penetrate the main surface. If the symmetry "type" format is used, the main normal at the node on the symmetry plane will be corrected to lie along the symmetry plane and contact will be enforced on the tangent plane as shown in *Figure 5*. Defining a YSYMM "type" boundary condition at node 1 to specify the symmetry plane will allow secondary node 100 to see the main surface `CSURF`.

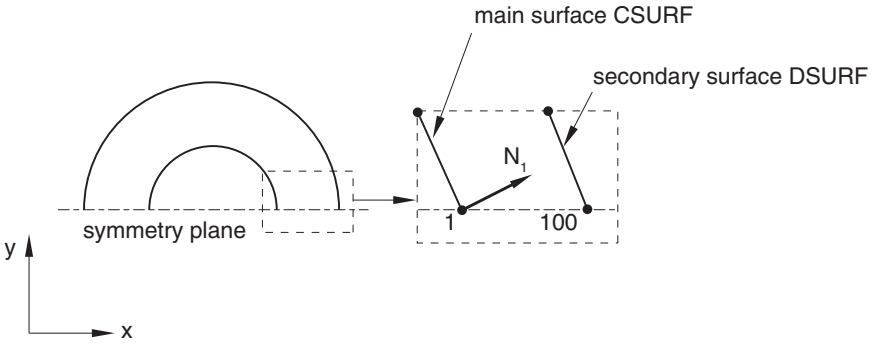


Figure 4: Main surface normal at node 1 in a small-sliding model of concentric cylinders. With the default $\mathbf{N_1}$ secondary node 100 will never contact `CSURF`.

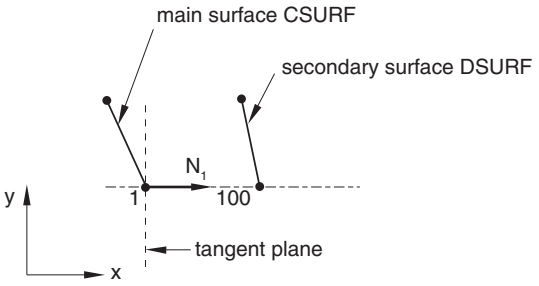


Figure 5: The modified main surface normal at node 1 of `CSURF` now allows secondary node 100 to contact `CSURF`.

- Modifying the local tangent plane orientation: In some cases the contact direction, $\mathbf{N}(\mathbf{X_0})$, defined from the main surface averaged normals will not define the contact surface accurately. The most common example of this is a circular surface meshed with nonuniform length facets. *Figure 6* shows how the averaged main normals will not be oriented correctly in the radial direction.

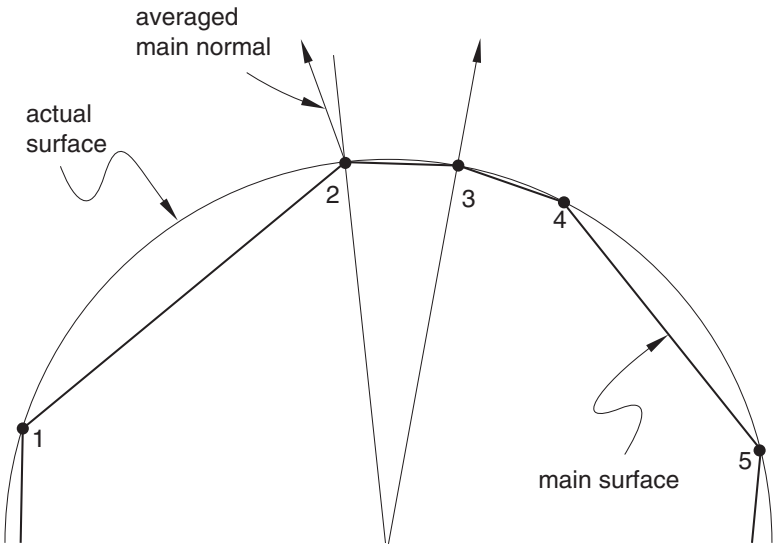


Figure 6: Poorly oriented averaged main surface normals for an irregularly meshed circular surface.

In this case you should specify the contact direction directly for each secondary node by defining spatially varying initial clearances (see *Specifying Initial Clearance Values Precisely*). The location of the anchor point is not affected by reorienting the tangent plane using an initial clearance definition.

Local Tangent Plane Rotation

The local tangent plane is always orthogonal to the contact direction. The contact direction is taken as the interpolated normal of the main surface at the anchor point, $\mathbf{N}(\mathbf{X_0})$, or as the direction specified with a spatially varying clearance definition (see *Specifying Initial Clearance Values Precisely*). Once the contact direction has been defined, the orientation of the local tangent plane with respect to the main surface facet remains fixed. Because the small-sliding formulation considers nonlinear geometric effects, Abaqus/Explicit continuously updates the orientation of the local tangent plane to account for the rotation of the main surface facet. The position of the anchor point relative to the surrounding nodes on the main surface facet does not change as the main surface deforms.

Load Transfer

In a small-sliding analysis the secondary node will transfer load to the nodes of the main surface facet containing the anchor point, with the magnitude of the load transferred to each node weighted by its proximity to the anchor point. For example, in *Figure 3* node 103 transmits load to both nodes 2 and 3 on the main surface. Thus, if node 103 impacts the local tangent plane, a larger share of the force would be transmitted to node 3 because it is closer to the anchor point $\mathbf{X_0}$.

As a secondary node slides along its local tangent plane, Abaqus/Explicit does not update the distribution of load transferred by a given secondary node to its associated main surface nodes; the distribution is based solely on the position of the anchor point. This is unlike the small-sliding formulation in Abaqus/Standard, which does update the load distribution to the main surface nodes as sliding occurs, so that no net moment is associated with the contact forces acting on secondary and main nodes per active contact constraint, regardless of the amount of sliding. Some net moment will be associated with the contact forces after sliding has occurred with the small-sliding formulation in Abaqus/Explicit. This net moment will not be significant if the sliding is truly small compared to element dimensions, but otherwise it can result in non-physical behavior and poor accounting of energy.

Figure 7 shows the potential problem that arises if small sliding is used but the relative tangential motion of the surfaces is not "small."

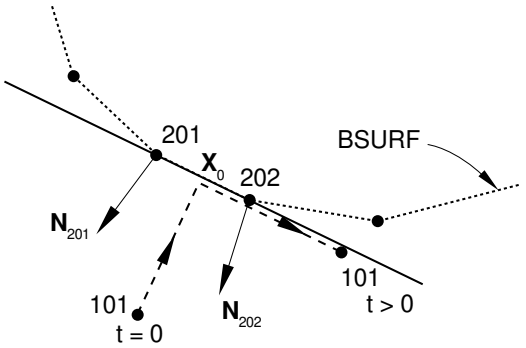


Figure 7: Excessive sliding in a small-sliding contact analysis.

It shows the possible evolution of contact between secondary node 101 in *Figure 1* and its main surface `BSURF`. Using the unit normal vectors $\mathbf{N}_{201}$ and $\mathbf{N}_{202}$, the anchor point $\mathbf{X}_0$ was found for secondary node 101; for the purposes of this example, assume that it lies at the midpoint of the 201–202 face. With this location of $\mathbf{X}_0$ the local tangent plane for node 101 is parallel with the 201–202 face. The load transfer always occurs at the original anchor point between nodes 201 and 202, no matter how far node 101 has slid along the local tangent plane. Therefore, if node 101 moves as shown in *Figure 7*, it will continue to transmit load equally to nodes 201 and 202 when, in fact, it really slid off the mesh forming the main surface `BSURF`.

What Can Be Considered Small Sliding

A contact pair in a small-sliding contact simulation should not grossly violate any of the assumptions or limitations outlined above. Adhere to the following guidelines:

- Secondary nodes should slide less than an element length from their corresponding anchor point and still be contacting their local tangent plane. If the main surface is highly curved, the secondary nodes should slide only a fraction of an element length.
- The local tangent planes formed by Abaqus/Explicit should be a good approximation of the mesh geometry; if necessary, use an initial clearance definition (*Specifying Initial Clearance Values Precisely*) to improve the tangent plane orientation.
- The rotation and deformation of the main surface should not cause the local tangent planes to become a poor representation of the main surface during the course of the analysis.

Main Surface Refinement in Small-Sliding Problems

The basic guidelines for pure main-secondary contact given previously in this section should still be followed in a small-sliding simulation. However, in a small-sliding simulation more thought must be given to the degree of refinement for the main surface.

The smoothly varying main surface normal $\mathbf{N}(\mathbf{x})$ and the local tangent planes that are formed with it are crucial to the success of a small-sliding analysis. As has been mentioned previously, there are several methods that can be used to modify $\mathbf{N}(\mathbf{x})$; however, they only control the initial configuration of the local tangent planes. The deformation and rotation of the main surface can reorient the local tangent planes such that they become a poor representation of the main surface. *Figure 8* shows an example where distortion of the main surface results in such a situation.

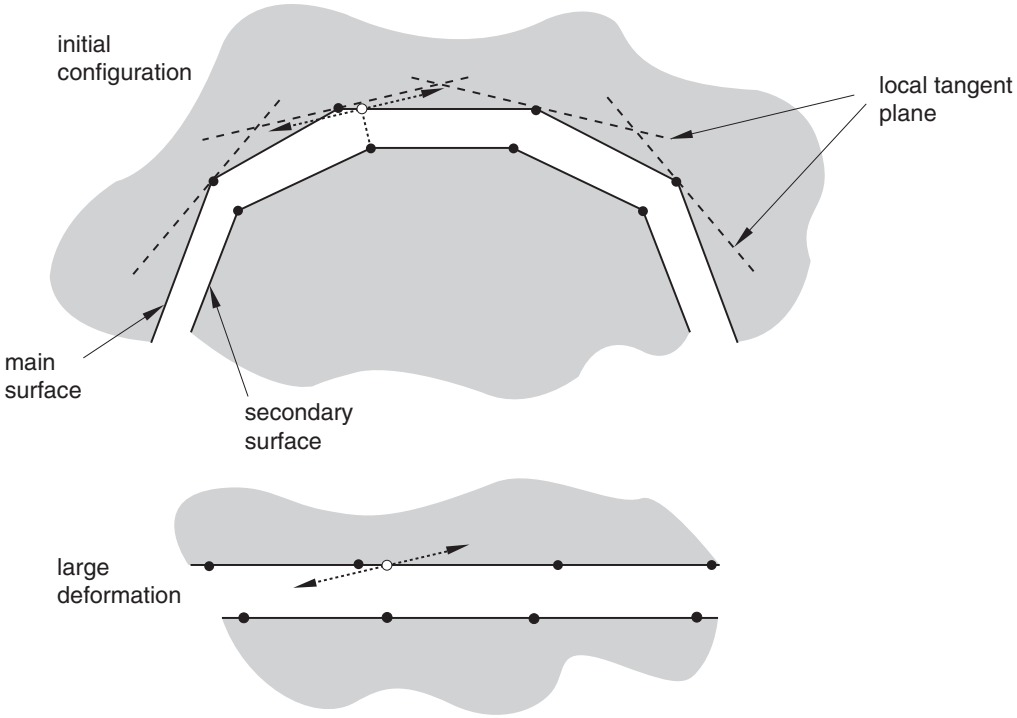


Figure 8: Main surface deformation in a small-sliding contact analysis can cause problems with the local tangent planes.

This problem can be minimized to some extent by using a more refined mesh on the main surface, thus providing more element faces to control the motion of the tangent planes. Excessive mesh refinement should not be necessary since only small sliding should occur.

Using the Infinitesimal-Sliding Formulation

The difference between the infinitesimal-sliding and small-sliding formulations is that the infinitesimal-sliding formulation ignores nonlinear geometric effects. To specify the infinitesimal-sliding formulation, you choose the small-sliding contact formulation and a small-displacement formulation for the analysis step.

Infinitesimal sliding assumes that both the relative motions of the surfaces and the absolute motions of the model remain small. The orientations of the local tangent planes are not updated, and the load transfer paths and the weightings assigned to each main surface node remain constant during an infinitesimal-sliding simulation.

Input File Usage: Use both of the following options:

**STEP*, NLGEOM=NO
…
**CONTACT PAIR*, SMALL SLIDING

Abaqus/CAE Usage: Interaction module: interaction editor: **Sliding formulation: Small sliding**
Step module: step editor: **Nlgeom: Off**

Local Tangent Directions for Contact

Local tangent directions for contact provide a reference frame for select contact pair output variables in Abaqus/Explicit (see *About General Contact in Abaqus/Explicit*). These local tangent directions are separate

from local coordinate systems associated with user subroutines *VFRIC* and *VUINTER*. Abaqus/Explicit establishes and updates the orientation of the first local contact tangent direction, $\mathbf{t_1}$, at secondary nodes according to the logic described below for different contact formulation types within a contact pair. The orientation of the second local tangent direction, $\mathbf{t_2}$, is found as the cross product of the contact normal direction, $\mathbf{n}$ with $\mathbf{t_1}$.

Contact pairs not involving an analytical surface

The $\mathbf{t_1}$-direction at a secondary node is re-established each increment using the standard convention for calculating a first local surface tangent direction (see *Conventions*). Local tangent directions do not co-rotate with the secondary surface in this case, which is unlike general contact (see *Local Tangent Directions for Contact*).

Contact pairs involving an analytical surface

The $\mathbf{t_1}$-direction for contact is initialized at a secondary node upon first contact to be aligned with the convention for the $\mathbf{t_1}$-direction of the analytical rigid surface discussed in *Analytical Rigid Surface Definition* at the point of contact. In subsequent increments, the $\mathbf{t_1}$-direction for contact at a secondary node evolves such that it continues to be aligned with the $\mathbf{t_1}$-direction of the analytical rigid surface at the current point of contact.

Contact Tracking Algorithms

A large portion of the computational cost associated with Abaqus/Explicit contact pairs derives from the algorithms used to track the relative motion between two contacting surfaces. There are two tracking approaches for the contact pair algorithm in Abaqus/Explicit, depending on the sliding formulation that is used: finite sliding and small/infinitesimal sliding.

Finite-Sliding Tracking

Abaqus/Explicit is designed to simulate highly nonlinear events or processes. Because it is possible for a node on one surface to contact any of the facets on the opposite surface, Abaqus/Explicit must use sophisticated search algorithms for tracking the motions of the surfaces.

The contact search algorithm is designed to be robust, yet computationally efficient. This algorithm assumes that the incremental relative tangential motion between surfaces does not significantly exceed the dimensions of the main surface facets, but there is no limit to the overall relative motion between surfaces. It is rare for the incremental motion to exceed the facet size because of the small time increment used in explicit dynamic analyses. In cases involving relative surface velocities that exceed material wave speeds, it may be necessary to reduce the time increment.

The contact search algorithm uses a global search at the beginning of each step, and a hierarchical global/local search algorithm is used for the other increments. The default contact search algorithm can handle the majority of typical contact situations. However, there are some situations that require special attention. We will consider a pure main-secondary contact pair for discussion purposes. For a balanced main-secondary contact pair, the contact search computations are performed twice for each contact pair.

Global Contact Searches

A global search determines the globally nearest main surface facet for each secondary node in a given contact pair. A bucket sorting algorithm is used to minimize the computational expense of these searches. A two-dimensional example, without consideration of “buckets,” is shown in *Figure 9*.

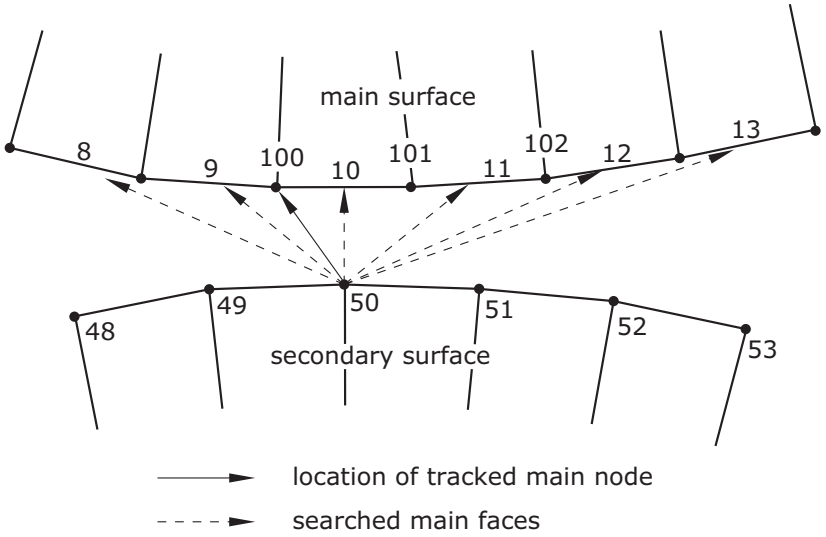


Figure 9: Global search in two dimensions.

The global search computes the distance from node 50 to all of the main surface facets in the same bucket as node 50. It determines that the nearest facet on the main surface to node 50 is the facet of element 10. Node 100 is the node on this facet that is nearest to node 50, and it is designated the tracked main surface node. This search is conducted for each secondary node, comparing each node against all of the facets on the main surface that are in the same bucket.

By default, Abaqus/Explicit performs a global search every one hundred increments for two-surface contact pairs. The frequency of the global search can be manually adjusted, as discussed in *Contact Controls for Contact Pairs in Abaqus/Explicit*. Despite the bucket sorting algorithm, global searches are computationally expensive: performing a global contact search in every increment will more than double the run time of many Abaqus/Explicit contact analyses.

Local Contact Searches

Abaqus/Explicit uses a local contact search to track the motion of the surfaces during most increments of an analysis. In this approach a given secondary node searches only the facets that are attached to the previously tracked main surface node. Abaqus/Explicit determines which adjacent facet is the nearest to the secondary node. It then determines which node on that facet is the closest main surface node to the secondary node and updates the tracked main surface node. If the closest main surface node is not the same as the previously tracked main surface node, Abaqus/Explicit performs another iteration of the local search.

In the example shown in *Figure 10*, node 50 moves as shown during an increment. In the first iteration of the search Abaqus/Explicit finds that the main surface facet on element 10 is still the closest facet of those attached to node 100 but that node 101 is now the tracked main surface node. Because the previously tracked node was node 100, Abaqus/Explicit performs another iteration. In this second iteration a new element, element 11, is found to be the closest facet and the closest main surface node is 102. Another iteration is performed because the identity of the tracked main surface node changed. In the third iteration the identity of the tracked node does not change, so Abaqus/Explicit designates node 102 as the tracked main surface node for secondary node 50.

A local search is substantially less expensive computationally than a global search. A slightly more expensive local search algorithm can be employed in situations where contact is not being properly enforced; this alternate algorithm is discussed in *Contact Controls for Contact Pairs in Abaqus/Explicit*.

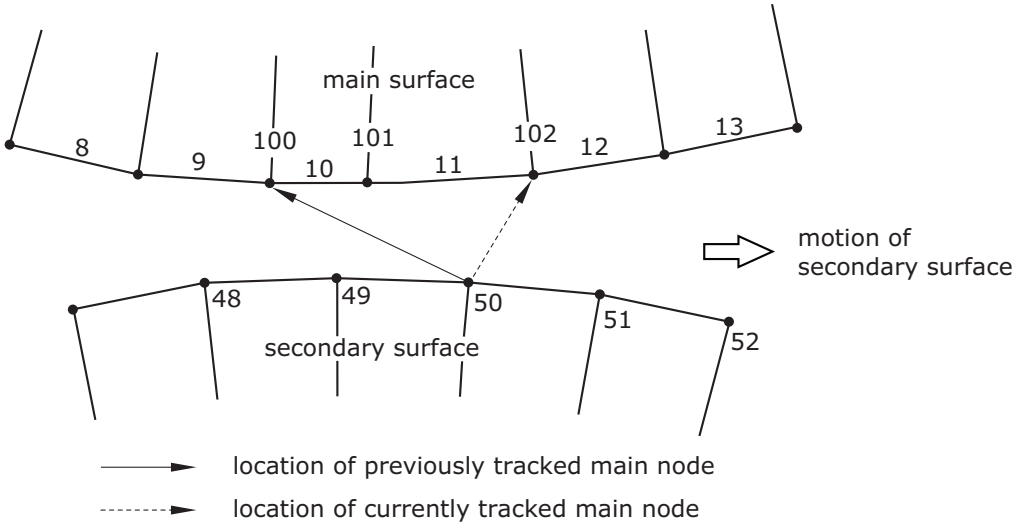


Figure 10: Local search in two dimensions.

Tracking Approach for Self-Contact Pairs

Abaqus/Explicit uses similar contact searching methods for simulations with self-contact as for two-surface contact; however, more frequent global searches are often necessary for self-contact problems. By default, contact pairs with self-contact use a global contact search every four increments, compared to every 100 increments for two-surface contact pairs; the frequency of the global searches can be manually adjusted (see *Contact Controls for Contact Pairs in Abaqus/Explicit*). If several facets that are unconnected to each other are found to be near a secondary node during global tracking, global tracking automatically will be performed more frequently than the specified number of increments. Despite this precaution, the self-contact algorithm will be less robust if you specify a search frequency that is significantly lower than the default.

Small-Sliding (or Infinitesimal-Sliding) Tracking Approach

When the small-sliding or infinitesimal-sliding contact approach is invoked (see *Sliding Formulation*), Abaqus/Explicit performs a single global search at the beginning of the first step to determine the globally nearest main surface facet for each secondary node in the given contact pair. Once the nearest facet has been determined, the nearest point on that facet defines the anchor point. Contact constraints will not be applied to secondary nodes that do not project onto any main surface facet. No further tracking is performed during the step or for subsequent steps in which the contact pair remains active. This makes the small-sliding/infinitesimal-sliding contact approach less expensive computationally than the finite-sliding contact approach. The cost savings are most significant for three-dimensional contact problems.

Contact Constraint Enforcement Methods in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *About General Contact in Abaqus/Explicit*
- *About Contact Pairs in Abaqus/Explicit*
- **CONTACT*
- **CONTACT PAIR*
- *Specifying contact formulation assignments for general contact*

Overview

Abaqus/Explicit uses two different methods to enforce contact constraints:

- The kinematic contact algorithm uses a kinematic predictor/corrector contact algorithm to strictly enforce contact constraints (for example, no penetrations are allowed).
- The penalty contact algorithm has a weaker enforcement of contact constraints but allows for treatment of more general types of contact.

Contact pairs in Abaqus/Explicit use kinematic enforcement by default, but penalty enforcement can be specified for individual contact pairs. General contact always uses penalty enforcement. Both methods conserve momentum between the contacting bodies.

Kinematic Contact Algorithm

A summary of the default kinematic algorithm that Abaqus/Explicit uses to enforce contact with the contact pair algorithm is presented below. It is a predictor/corrector algorithm and, therefore, has no influence on the stable time increment. It is easier to describe the algorithm by first considering a pure main-secondary contact pair.

Kinematic Enforcement of Contact Conditions in a Pure Main-Secondary Contact Pair

In this case in each increment of the analysis Abaqus/Explicit first advances the kinematic state of the model into a predicted configuration without considering the contact conditions. Abaqus/Explicit then determines which secondary nodes in the predicted configuration penetrate the main surfaces. The depth of each secondary node's penetration, the mass associated with it, and the time increment are used to calculate the resisting force required to oppose penetration. For hard contact, this is the force which, had it been applied during the increment, would have caused the secondary node to exactly contact the main surface. The next step depends on the type of main surface used.

- When the main surface is formed by element faces, the resisting forces of all the secondary nodes are distributed to the nodes on the main surface. The mass of each contacting secondary node is also distributed to the main surface nodes and added to their mass to determine the total inertial mass of the contacting interfaces. Abaqus/Explicit uses these distributed forces and masses to calculate an acceleration correction for the main surface nodes. Acceleration corrections for the secondary nodes are then determined using the predicted penetration for each node, the time increment, and the acceleration corrections for the main surface nodes. Abaqus/Explicit uses these acceleration corrections to obtain a corrected configuration in which the contact constraints are enforced.

- In the case of an analytical rigid main surface, the resisting forces of all secondary nodes are applied as generalized forces on the associated rigid body. The mass of each contacting secondary node is added to the rigid body to determine the total inertial mass of the contacting interfaces. The generalized forces and added masses are used to calculate an acceleration correction for the analytical rigid main surface. Acceleration corrections for the secondary nodes are then determined by the corrected motion of the main surface.

When using hard kinematic contact, it is still possible with the pure main-secondary algorithm for the main surface to penetrate the secondary surface in the corrected configuration (see *Figure 1*).

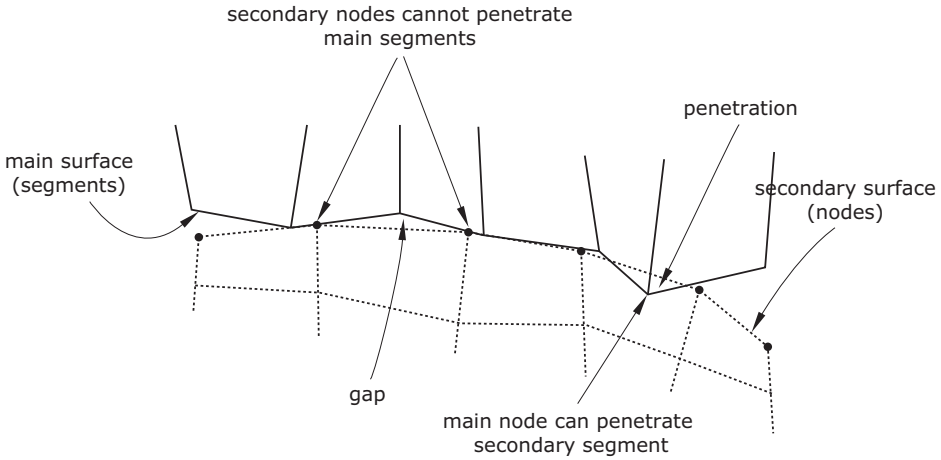


Figure 1: Main surface penetrations into the secondary surface of a pure main-secondary contact pair due to coarse discretization.

Using a sufficiently refined mesh on the secondary surface will minimize such penetrations. Softened kinematic contact will allow penetrations since corrections are made to satisfy the pressure-overclosure relationship at the secondary nodes, not the condition of zero penetration.

Kinematic Enforcement of Contact Conditions in a Balanced Main-Secondary Contact Pair

The kinematic contact algorithm for a balanced main-secondary contact pair applies acceleration corrections that are linear combinations of pure main-secondary corrections calculated in exactly the same manner as outlined above. One set of corrections is calculated considering one surface as the main surface, and the other corrections are calculated considering that same surface as the secondary surface. Abaqus/Explicit then applies a weighted average of the two values. The exact weighting for each correction depends on the weighting factor specified for the contact pair (see *Contact Surface Weighting*). The default for balanced main-secondary contact is to weight each correction equally.

Hard kinematic contact will minimize the penetration of the surfaces. However, after the initial weighted correction is applied, it is possible to still have some penetration of the surfaces. Therefore, Abaqus/Explicit uses a second contact correction to resolve any remaining overclosure in a balanced main-secondary contact pair that uses hard kinematic contact. Both main-secondary assignment combinations are again considered, but weighting factors are not used when combining the contributions to form the second applied acceleration correction. It is possible that small gaps between the contacting surfaces will be created during the second correction if there was some residual penetration after the first correction: the magnitude of the gaps after the second correction will generally be much smaller than the penetration after the first correction. The effect of the second correction is illustrated in *Figure 2* to *Figure 5*.

The second contact correction described above is not conducted in the case when a softened kinematic contact formulation is used. This may lead to penetration values that may not be exactly synchronized with the pressure-overclosure curve. Moreover, the frictional shear forces (if any) may not reflect the specified coefficient of friction exactly when non-sticking sliding occurs. Use a pure main-secondary kinematic formulation to avoid these inaccuracies.

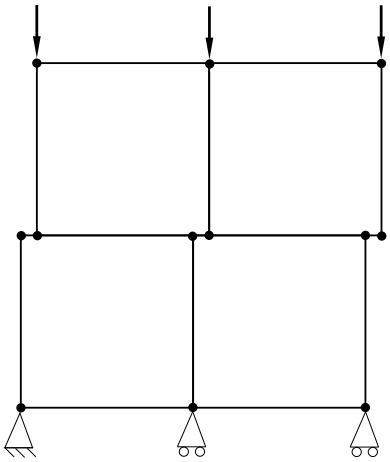


Figure 2: Effect of second contact corrections; initial configuration.

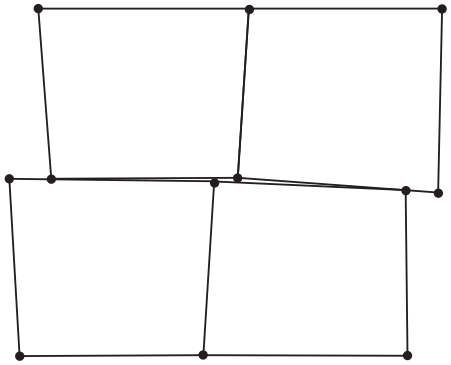


Figure 3: Final configuration when the second contact correction is used.

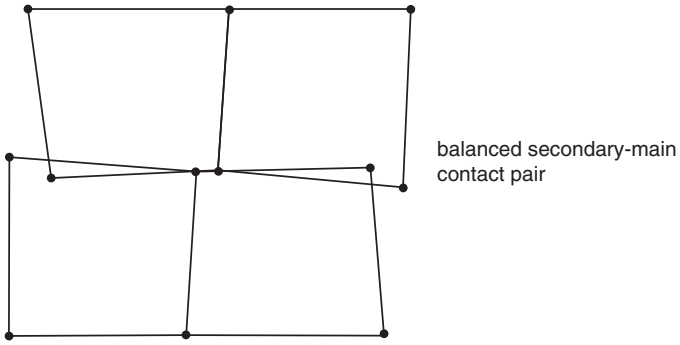


Figure 4: Final configuration if the second contact correction were to be omitted.

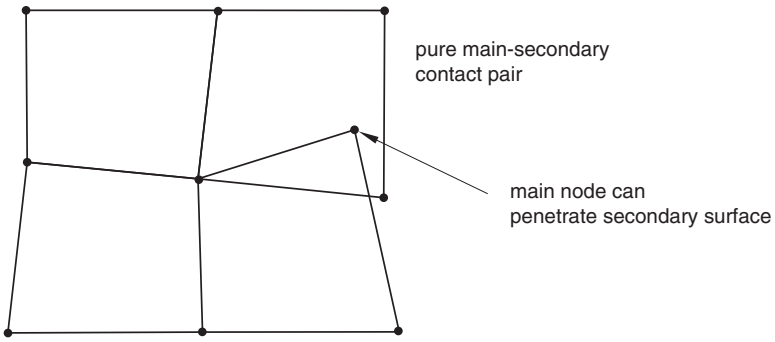


Figure 5: Final configuration when a pure main-secondary contact pair is used. The main surface is defined on the bottom elements.

Energy Considerations for Hard Kinematic Contact

The kinematic contact algorithm strictly enforces contact constraints and conserves momentum. To achieve these qualities with a discretized model, some energy is absorbed upon impact. For example, consider a linear elastic beam modeled with several elements that impacts a rigid wall as shown in *Figure 6*. The kinetic energy of the leading node is absorbed by the contact algorithm upon impact. A stress wave passes through the truss, and the truss eventually rebounds from the wall. The kinetic energy after the rebound is smaller than before the impact because of the contact node's energy loss upon impact. As the mesh is refined, this energy loss is reduced because the mass and kinetic energy of the leading node of the truss become less significant.

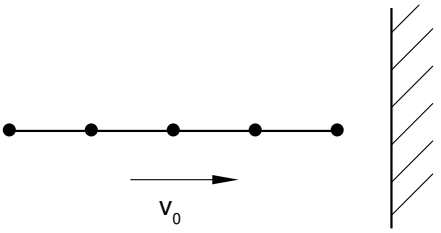


Figure 6: Beam impacting a fixed rigid wall.

Contact forces can also exert negative external work upon impact since contact forces act over the entire increment in which impact occurs, including the fraction of the increment prior to impact. The opposing contact forces, which are equal in magnitude, act over different distances, thereby exerting a nonzero net work. The net external work of these forces is negative, and the absolute value of the net external work does not exceed the contact node's kinetic energy loss upon impact. These energies are insignificant in most models but can be significant in high-speed impacts, where high mesh refinement near the contact interface is recommended.

Penalty Contact Algorithm

The penalty contact algorithm results in less stringent enforcement of contact constraints than the kinematic contact algorithm, but the penalty algorithm allows for treatment of more general types of contact (for example, contact between two rigid bodies). The penalty contact method is well suited for very general contact modeling, including the following situations:

- multiple contacts per node,
- contact between rigid bodies, and
- contact of surfaces also involved in other types of constraints (such as MPCs).

Since the penalty algorithm introduces additional stiffness behavior into a model, this stiffness can influence the stable time increment. Abaqus/Explicit automatically accounts for the effect of the penalty stiffnesses in the automatic time incrementation, although this effect is usually small, as discussed below.

The penalty enforcement method is always used by the general contact algorithm. For contact pairs, you can specify the penalty method as an alternative to the default kinematic enforcement method. When the penalty method is chosen for enforcing contact constraints in the normal direction, it is also used to enforce sticking friction (see *Frictional Behavior*).

Input File Usage: Use the following option to select the penalty contact algorithm for a contact pair:

**CONTACT PAIR*, MECHANICAL CONSTRAINT=PENALTY
surface_1, *surface_2*

Abaqus/CAE Usage: Interaction module: interaction editor: **Mechanical constraint formulation: Penalty contact method**

Penalty Enforcement of Contact Conditions for Pure Main-Secondary Surface Weighting

The penalty contact algorithm searches for secondary node penetrations in the current configuration, including node-into-face, node-into-analytical rigid surface, and edge-into-edge penetrations. For node-to-face contact, forces that are a function of the penetration distance are applied to the secondary nodes to oppose the penetration, while equal and opposite forces act on the main surface at the penetration point. The main surface contact forces are distributed to the nodes of the main faces being penetrated. For node-to-analytical rigid surface contact, forces that are a function of the penetration distance are applied to the secondary nodes to oppose the penetration, while equal and opposite forces act on the analytical rigid surface at the penetration point. The contact forces acting at the penetration point of the analytical rigid surface result in equivalent forces and moments at the reference node of the rigid body corresponding to the analytical rigid surface. For edge-to-edge contact, the opposing contact forces are distributed to the nodes of the two contacting edges.

As with the pure main-secondary kinematic contact algorithm, there is no resistance to main surface nodes penetrating secondary surface faces with the pure main-secondary penalty contact algorithm. Using a sufficiently refined mesh on the secondary surface will help correct this problem.

Penalty Enforcement of Contact Conditions for Balanced Main-Secondary Surface Weighting

The penalty contact algorithm for balanced main-secondary contact surfaces computes contact forces that are linear combinations of pure main-secondary forces calculated in the manner outlined above. One set of forces is calculated considering one surface as the main surface, and the other forces are calculated considering that same surface as the secondary surface. Abaqus/Explicit then applies a weighted average of the two values. The weighting used with each set of forces depends on the weighting factor specified for the surfaces (see *Contact Formulation for General Contact in Abaqus/Explicit* and *Contact Formulations for Contact Pairs in Abaqus/Explicit*). The default for balanced main-secondary contact pairs and general contact is to weight each of the two sets of forces equally.

Constant or Nonlinear Penalty Stiffness

The penalty stiffness is typically a constant "spring" stiffness that relates the contact pressure or force to the penetration distance. However, nonlinear contact pressure-overclosure behavior occurs by default for contact involving low-density foam and elastomeric foam materials. For these cases, the slope of the contact pressure-overclosure curve increases after an underlying foam material stiffens significantly under compression. The nonlinear contact pressure-overclosure curve evolves as penetration increases and remains in effect as penetration decreases. Therefore, the net work done by normal contact forces (aside from contact damping

effects) is approximately zero after a complete contact cycle from initial touching to contact opening, regardless of whether the penalty stiffness is constant or nonlinear.

Scaling the Penalty Stiffness

The constant or nonlinear penalty stiffness curve is chosen automatically by Abaqus/Explicit for hard penalty contact, such that the effect on the time increment is minimal yet the allowed penetration is not significant in most analyses. The default penalty stiffness is based on a representative stiffness of the underlying elements. A scale factor is applied to this representative stiffness to set the default penalty. Consequently, the penetration distance will typically be greater than the parent elements' elastic deformation normal to the contact interface. In purely elastic problems this penetration can affect the stress solution significantly, as demonstrated in *The Hertz contact problem*.

When element or node-based rigid bodies are involved in contact interactions, for numerical stability reasons Abaqus/Explicit will compute penalties at each contacting node on the rigid body by considering the overall inertia properties of the body. Consequently, the contact penalties will be different from the case when these elements were not converted to rigid and, thus, the penetrations in the two cases may be different.

You can specify a factor by which to scale the default constant or nonlinear penalty stiffnesses, as described in *Contact Controls for General Contact in Abaqus/Explicit* and *Contact Controls for Contact Pairs in Abaqus/Explicit*. This scaling may affect the automatic time incrementation. Use of a large scale factor is likely to increase the computational time required for an analysis because of the reduction in the time increment that is necessary to maintain numerical stability.

Choosing between the Kinematic and Penalty Contact Algorithms

The penalty contact algorithm can model some types of contact that the kinematic contact algorithm cannot. Element-based rigid surfaces are not restricted to acting only as main surfaces within the penalty algorithm as they are within the kinematic algorithm. Thus, the penalty method allows modeling of contact between rigid surfaces, except when both surfaces are analytical rigid surfaces or when both surfaces are node-based.

The penalty contact algorithm must be used for all contact pairs involving a rigid body if a linear constraint equation, multi-point constraint, surface-based tie constraint, or connector element is defined for a node on the rigid body. For all other cases, Abaqus/Explicit enforces equations, multi-point constraints, tie constraints, embedded element constraints, and kinematic constraints (defined using connector elements) independently of contact constraints; therefore, if a degree of freedom participates in a linear constraint equation, multi-point constraint, tie constraint, embedded element constraint, or kinematic constraint in addition to a contact constraint, the contact constraint will usually override these constraints (see the discussion in *Conflicts with Multi-Point Constraints*). Hence, the penalty contact algorithm is recommended if these constraints need to be strictly enforced.

Impact is plastic when the default hard, kinematic contact algorithm is used and the kinetic energy of the contacting nodes is lost. This loss in energy is insignificant for a refined mesh but can be significant with a coarse mesh. Penalty contact and softened kinematic contact introduce numerical softening to the contact enforcement analogous to adding elastic springs to the contact interface, which means that these algorithms do not dissipate energy upon impact (the energy stored in the springs is recoverable). This distinction between the algorithms is particularly apparent if a point mass with no force acting upon it impacts a fixed rigid wall: with penalty contact and softened kinematic contact the point mass will bounce away, but with hard kinematic contact the point mass will stick to the wall.

A further difference between kinematic and penalty contact is that the critical time increment is unaffected by kinematic contact but can be affected by penalty contact. For hard penalty contact, default penalty stiffnesses are chosen such that the stable time increments of the deformable parent elements of contact surface facets are effectively reduced by approximately 4% for increments in which contact forces are being transmitted; default penalty stiffnesses of node-based surface nodes require a 1% decrease in the element-by-element time increment to ensure numerical stability. Penalty stiffnesses between rigid bodies are chosen by default to have no effect

on the stable time increment. If the default penalty stiffnesses are overridden by a penalty scale factor or softened contact behavior (see *Contact Pressure-Overclosure Relationships*), the time increment is modified based on the maximum stiffness active in the contact interface. Increasing the penalty stiffnesses may decrease the stable time increment significantly (see *Table 1*). If the overall stable time increment is not controlled by elements on the contact interface, the penalty contact algorithm usually will not affect the time increment.

Penalty contact and softened kinematic contact cannot be used with the breakable bond model; hard kinematic contact must be used for this model.

Table 1: Effect of scale factor on time increment.

Penalty scale factor	Lower bound to ratio of the time increment with contact divided by the time increment without contact
1.0	0.96
10.0	0.34
100.0	0.13
1000.0	0.04
10000.0	0.013

Contact Controls for General Contact in Abaqus/Explicit

Products: Abaqus/Explicit

References:

- *About General Contact in Abaqus/Explicit*
- *Assigning Surface Properties for Contact Pairs in Abaqus/Explicit*
- **CONTACT*
- **CONTACT CONTROLS ASSIGNMENT*

Overview

Contact controls for the general contact algorithm:

- can be used to selectively scale the default penalty stiffness for particular regions within a general contact domain;
- can be used to control whether nodes are removed from the general contact domain once all of the faces and edges to which they are attached have eroded;
- can be used to control whether checks need to be performed to prevent folds in general contact surfaces from inverting on themselves;
- can be used to modify the default initial overclosure resolution method for one or more pairs of surfaces in the general contact domain; and
- can be used to modify the default contact thickness reduction checks.

Scaling Default Penalty Stiffnesses

The general contact algorithm uses a penalty method to enforce the contact constraints (see *Contact Constraint Enforcement Methods in Abaqus/Explicit* for more information). The "spring" stiffness that relates the contact force to the penetration distance is chosen automatically by Abaqus/Explicit, such that the effect on the time increment is minimal yet the allowed penetration is not significant in most analyses. Significant penetrations may develop in an analysis if any of the following factors are present:

- Displacement-controlled loading
- Materials at the contact interface that are purely elastic or stiffen with deformation
- Deformable elements (especially membrane and surface elements) that have relatively little mass of their own and are constrained via methods other than boundary conditions (for example, connectors) involved in contact
- Rigid bodies that have relatively little mass or rotary inertia of their own and are constrained via methods other than boundary conditions (for example, connectors) involved in contact

See *The Hertz contact problem* for an example in which the first two of these factors combine such that the contact penetrations with the default penalty stiffness are significant.

You can specify a scale factor by which to modify penalty stiffnesses for specified interactions within the general contact domain. This scaling may affect the automatic time incrementation. Use of a large scale factor is likely to increase the computational time required for an analysis because of the reduction in the time increment that is necessary to maintain numerical stability (see *Contact Constraint Enforcement Methods in Abaqus/Explicit* for further discussion).

Optionally, you can introduce contact mass scaling to the surface nodes in contact to avoid a reduction in the time increment (for more information, see *Mass Scaling to Account for Contact Stiffness*).

The surface names used to specify the regions where nondefault penalty stiffness should be assigned do not have to correspond to the surface names used to specify the general contact domain. In many cases the contact interaction will be defined for a large domain, while a nondefault penalty stiffness will be assigned to a subset of this domain. If the surfaces to which a nondefault penalty stiffness is assigned fall outside the general contact domain, the controls assignment will be ignored. The last assignment will take precedence if the specified regions overlap.

Input File Usage: **CONTACT CONTROLS ASSIGNMENT*, TYPE=SCALE PENALTY
surface_1, *surface_2*, *scale_factor*

This option must be used in conjunction with the **CONTACT* option. It should appear at most once per step; the data line can be repeated as often as necessary to assign penalty stiffness scale factors to different regions. If the first surface name is omitted, a default surface that encompasses the entire general contact domain is assumed. If the second surface name is omitted or is the same as the first surface name, the specified contact controls are assigned to contact interactions between the first surface and itself. Keep in mind that surfaces can be defined to span multiple unattached bodies, so self-contact is not limited to contact of a single body with itself.

Control of Nodal Erosion

You can control whether contact nodes remain in the contact domain after all the surrounding faces and edges have eroded due to element failure. By default, these nodes remain in the contact domain and act as free-floating point masses that can experience contact with faces that are still part of the contact domain. You can specify that nodes of element-based surfaces should erode (i.e., be removed from the contact domain) once all contact faces and contact edges to which they are attached have eroded. Nodes that you include in the contact domain only with node-based surfaces are never removed from the contact domain.

Computational cost can increase as a result of free-flying nodes if nodal erosion is not specified, particularly for analyses conducted in parallel. The increased computational cost is related to the likelihood of free-flying nodes moving far away from the elements that remain active, which stretches the volume of the contact domain and thereby tends to increase contact search costs as well as the cost of communication between processors in parallel analysis. However, contact involving free-flying nodes can contribute significant momentum transfer in some cases, which will not be accounted for if nodal erosion is specified.

Input File Usage: **CONTACT CONTROLS ASSIGNMENT*, NODAL EROSION=NO

This option must be used in conjunction with the **CONTACT* option. This parameter setting applies to the entire general contact domain.

Activating the Fold Inversion Check

If a general contact surface contains sharp folds, significant loading events (for example, those encountered during the inflation of a folded airbag) may cause one or more of the folds to invert. Inversion is most likely to occur at a fold where edge-to-edge contact has not been activated on the edges of the faces forming the fold. The presence of edge-to-edge constraints usually prevents a fold from inverting. Inversion of a fold, in the absence of edge-to-edge contact constraints, may induce errors in the node-to-face contact tracking algorithm and may result in a node that was being tracked on a face that forms part of an inverted fold getting “snagged” on the wrong side of the tracked face. To avoid such situations, it may be desirable to activate the fold inversion check for models containing sharp folds. The fold inversion check detects situations where a fold is about to invert and applies a force field to the faces forming the fold to prevent the fold from inverting.

The fold inversion check is activated on a surface-by-surface basis. You must specify the surface name for which the fold inversion check needs to be activated. If activated for a particular surface, the fold inversion check applies to all folds within that surface.

The surface names used to specify the regions where the fold inversion check should be activated do not have to correspond to the surface names used to specify the general contact domain. In many cases the contact interaction will be defined for a large domain, while the fold inversion check will be activated in a subset of this domain. If the surfaces for which the fold inversion check needs to be activated fall outside the general contact domain, the controls assignment is ignored.

Input File Usage: **CONTACT CONTROLS ASSIGNMENT*,
TYPE=FOLD INVERSION CHECK
surface_1

This option must be used in conjunction with the **CONTACT* option. It should appear at most once per step; the data line can be repeated as often as necessary to activate the fold inversion check in different regions of the contact domain. If the surface name is omitted, a default surface that encompasses the entire general contact domain is assumed.

Control of Initial Overclosure Resolution

By default, Abaqus/Explicit automatically adjusts the positions of surfaces to remove small initial overclosures that exist in the general contact domain in the first step of a simulation. Conflicting adjustments from separate contact definitions, boundary conditions, tie constraints, and rigid body constraints can cause incomplete resolution of initial overclosures. Initial overclosures that are not resolved by repositioning nodes are stored as initial contact offsets to avoid large contact forces at the beginning of an analysis.

Alternatively, in certain situations it may be desirable to avoid nodal adjustments altogether between a pair of surfaces and to treat all initial overclosures between the surfaces as temporary contact offsets. You can then specify the surfaces for which the initial overclosures should not be resolved by nodal adjustments and which should instead be stored as offsets.

Input File Usage: **CONTACT CONTROLS ASSIGNMENT*, AUTOMATIC OVERCLOSURE RESOLUTION
surface_1, *surface_2*, *STORE OFFSETS*

This option must be used in conjunction with the **CONTACT* option. It should appear at most once per step; the data line can be repeated as often as necessary to assign a nondefault overclosure resolution method to different regions. If the first surface name is omitted, a default surface that encompasses the entire general contact domain is assumed. If the second surface name is omitted or is the same as the first surface name, the specified contact controls are assigned to contact interactions between the first surface and itself.

Effect of Control of Initial Overclosure Resolution with Edge-to-Edge Interactions

Contact offsets are associated with individual node-facet and edge-edge combinations. Upon sliding, Abaqus/Explicit attempts to transfer contact offsets to different node-facet or edge-edge pairings, as appropriate. However, a contact offset may not be maintained (that is, may become zero) upon sliding for some cases involving multiple contacts for individual nodes or edges or surfaces with corners. Limitations causing discontinuities in the value of a contact offset across increments, which are more likely for edge-to-edge contact than node-to-surface contact, can locally degrade a solution, cause a solution to depend on the number of processors used, or cause

an analysis to exit. These limitations can be avoided by more careful positioning of surface nodes by your preprocessor or, in many cases, allowing strain-free adjustments to occur.

Control of Contact Thickness Reduction Checks

By default, the general contact algorithm requires that the contact thickness does not exceed a certain fraction of the surface facet edge lengths or diagonal lengths. This fraction generally varies from 20% to 60% based on the geometry of the element and whether the element is near a shell perimeter. The general contact algorithm will scale back the contact thickness automatically where necessary without affecting the thickness used in the element computations for the underlying elements.

To check whether the thickness needs to be reduced in any particular region in the model, the contact algorithm first assigns the full thickness to each contact node, represented by a sphere centered at the node with a diameter equal to the thickness. Next, the thickness is reduced so that the spheres do not overlap with any neighboring facets that are not attached directly to the node, preventing spurious self-contact from developing. Then, the nodes on the perimeter of shells are moved a maximum of 50% of the facet size in the plane of the facet away from the perimeter to eliminate the "bull-nose" effect that occurs with the contact pair algorithm (see *Assigning Surface Properties for Contact Pairs in Abaqus/Explicit*). If the thickness of the shell perimeter nodes is greater than twice the maximum perimeter offset, a final thickness reduction is performed to eliminate the remainder of the "bull-nose."

If the default thickness reductions are unacceptable in particular regions of the model, you can exclude self-contact for those regions via contact exclusion definitions (see *About General Contact in Abaqus/Explicit*) and activate a control for the contact thickness reduction checks.

Input File Usage: Use the following option to eliminate thickness reductions in regions of the model that are excluded from self-contact, while still reducing thickness at shell perimeters where perimeter offsets are insufficient to avoid the "bull-nose" effect:

```
*CONTACT CONTROLS ASSIGNMENT,
CONTACT THICKNESS REDUCTION=SELF
```

Use the following option to eliminate thickness reductions in regions of the model that are excluded from self-contact and at all shell perimeters (a "bull-nose" will form at shell perimeter nodes if the thickness is greater than twice the maximum perimeter offset):

```
*CONTACT CONTROLS ASSIGNMENT,
CONTACT THICKNESS REDUCTION=NOPERIMSELF
```

Consideration of Shell and Beam Thickness Offsets

During a contact analysis, the reference surface of shell and beam elements may be offset from the actual point of contact. Additional accuracy can be achieved by optionally accounting for offsets in slip computations and generating nodal contact moments such that the effective point of action of the contact force is at the desired location, as discussed in *Moment Associated with Frictional Force* and *Moment Associated with Normal Force*.

Input File Usage: Use the following option to account for offsets in slip computations and to generate nodal contact moments such that the effective point of action of the contact force is at the desired location:

```
*CONTACT CONTROLS ASSIGNMENT,
ROTATIONAL TERMS=STRUCTURAL
```

Use the following option (default) to ignore the effects of shell and beam thickness offsets:

**CONTACT CONTROLS ASSIGNMENT*,
ROTATIONAL TERMS=NONE

Moment Associated with Frictional Force

Figure 1 shows an example in which the non-default option to consider structural rotation terms should be activated to improve slip increment calculations (and, therefore, achieve proper enforcement of the sticking conditions) and to generate nodal contact moments to account for the fact that nodes are offset from the point of contact with a roller due to shell thickness.

As shown in *Figure 1*, some difference in tangential motion between the two reference surfaces should exist due to rotation of the thickness offset. A shell node in the sticking contact region should have slightly larger incremental displacement than that of the point of contact on the roller because the shell nodes are farther from the rotational axis, which will occur only if the non-default option to consider structural rotation terms in contact calculations is specified.

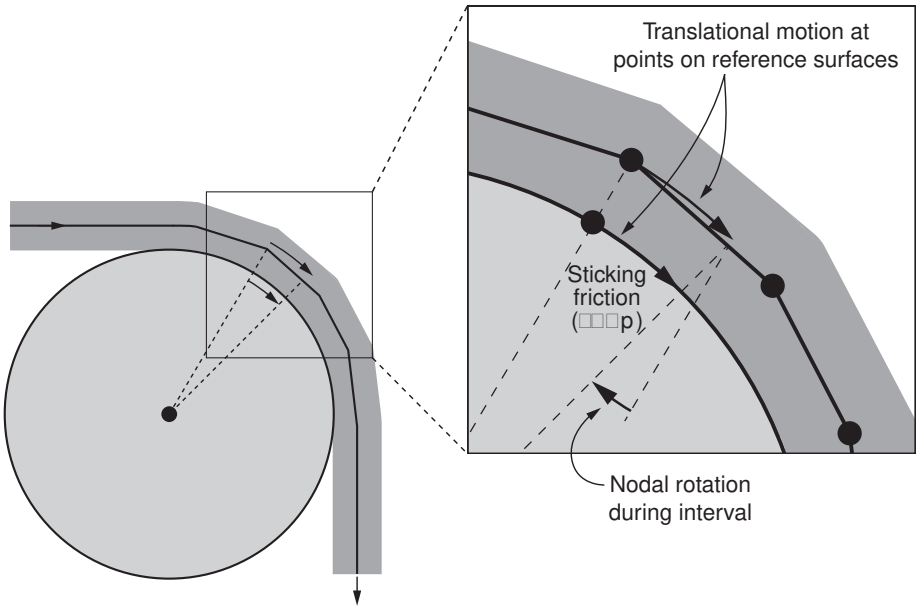


Figure 1: Effect of shell thickness on slip increment.

In the same example, applying a contact nodal moment together with the contact nodal forces at the shell node, as shown in *Figure 2*, causes the effective point of action of the contact force on the shell to act at the point of contact with the roller, such that this force directly opposes the contact force acting on the roller, as desired. Such contact nodal moments are generated only if the non-default option to consider structural rotation terms in contact calculations is specified.

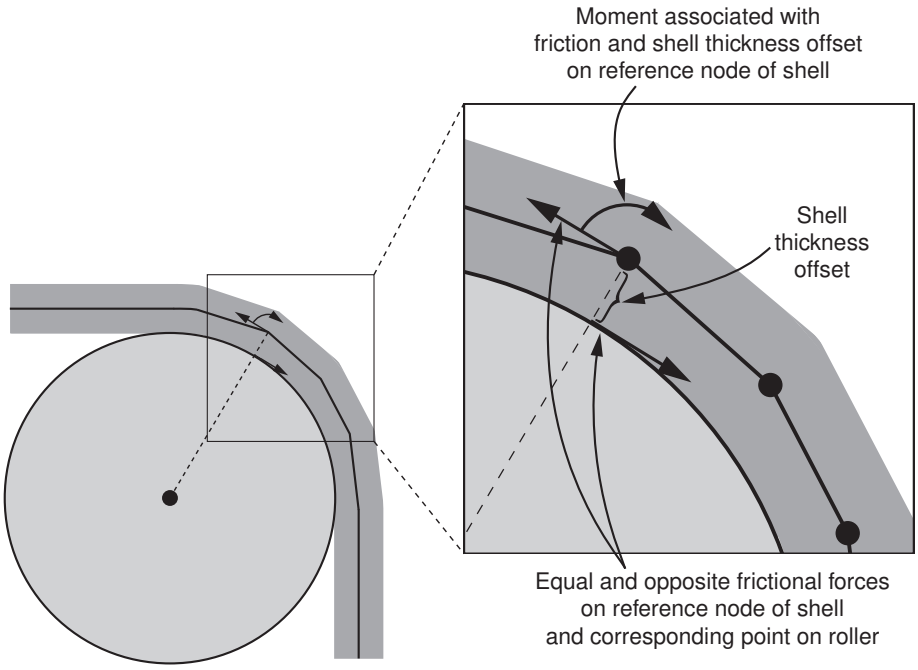


Figure 2: Nodal moment associated with frictional constraint.

Moment Associated with Normal Force

Figure 3 and *Figure 4* show another example in which it may be important to specify the non-default option to consider structural rotation terms in contact calculations. The contact nodal moment is associated with contact normal force and shell offset in this example. The center of action of the contact force acting on the body modeled with shell elements should be independent of whether the reference surface is offset from the center of the shell (see *Figure 3*). By default, the contact algorithm applies a nodal contact force without applying a nodal contact moment, as shown on the left side of *Figure 4*. However, with structural rotation terms accounted for in contact calculations, contact nodal moments are generated for the case with the reference surface offset from the midsurface, as shown on the right side of *Figure 4*, such that the effective point of the contact force acting on the shell (with combined effects of the nodal force and nodal moment) is at the desired location.

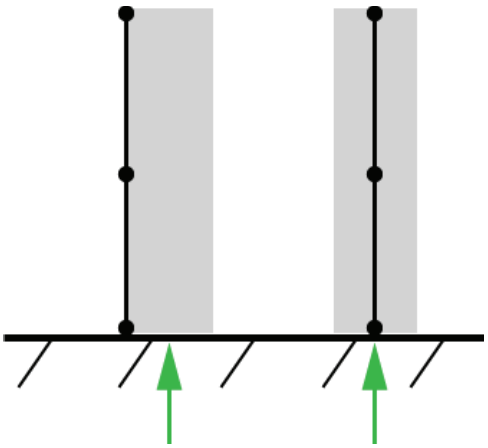


Figure 3: Ideal effective point of contact force with and without shell offset.

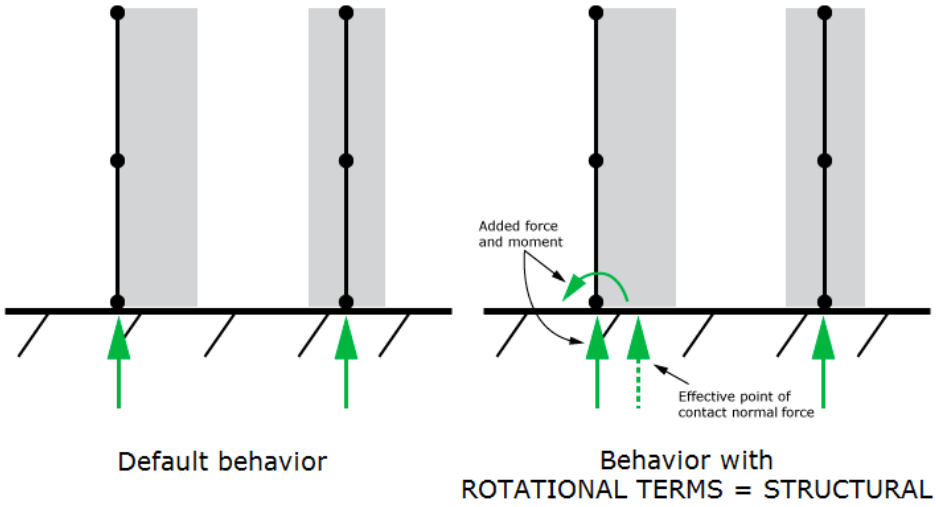


Figure 4: Effect of nodal moment in improving the effective point of contact for shell offset case.

Contact Controls for Contact Pairs in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *About Contact Pairs in Abaqus/Explicit*
- **CONTACT CONTROLS*
- *Specifying contact controls in an Abaqus/Explicit analysis*

Overview

Contact controls for Abaqus/Explicit contact pairs can be used

- to scale the stiffness used by penalty contact constraints, and
- to adjust the search algorithms that track the motions between two surfaces.

Scaling Default Penalty Stiffnesses

If you use the penalty method to enforce contact constraints in a contact pair (see *Contact Constraint Enforcement Methods in Abaqus/Explicit*), Abaqus/Explicit resists penetrations between surfaces by applying a "spring" stiffness to penetrating nodes. The "spring" stiffness that relates the contact force to the penetration distance is chosen automatically by Abaqus/Explicit, such that the effect on the time increment is minimal yet the allowed penetration is not significant in most analyses. Significant penetrations may develop in an analysis if any of the following factors are present:

- Displacement-controlled loading
- Materials at the contact interface that are purely elastic or stiffen with deformation
- Deformable elements (especially membrane and surface elements) that have relatively little mass of their own and are constrained via methods other than boundary conditions (for example, connectors) involved in contact
- Rigid bodies that have relatively little mass or rotary inertia of their own and are constrained via methods other than boundary conditions (for example, connectors) involved in contact

See *The Hertz contact problem* for an example in which the first two of these factors combine such that the contact penetrations with the default penalty stiffness are significant.

You can specify a scale factor by which to modify penalty stiffnesses for specified contact pairs. This scaling may affect the automatic time incrementation. Use of a large scale factor is likely to increase the computational time required for an analysis because of the reduction in the time increment that is necessary to maintain numerical stability (see *Contact Constraint Enforcement Methods in Abaqus/Explicit* for further discussion).

Input File Usage: Use both of the following options to scale the default penalty stiffnesses:

**CONTACT PAIR*, MECHANICAL CONSTRAINT=PENALTY,
CPSET=*contact_pair_set_name*
surface_1, *surface_2*
**CONTACT CONTROLS*, CPSET=*contact_pair_set_name*, SCALE
PENALTY=*factor*

Abaqus/CAE Usage: Interaction module:
Create Contact Controls: **Name:** *contact_controls_name*,
Abaqus/Explicit contact controls: **Penalty stiffness scaling factor:** *factor*

Interaction editor: **Mechanical constraint formulation: Penalty contact method**,
Contact controls: *contact_controls_name*

Adjusting the Finite-Sliding Contact Tracking Algorithm

In a finite-sliding contact pair, searches are conducted continually throughout an analysis to track the relative motion between the two contacting surfaces. The contact tracking algorithm consists of an expensive, periodic global search and a less expensive, regular local search; the search algorithms are discussed in detail in *Contact Tracking Algorithms*. You can use contact controls to adjust the frequency and cost of these searches.

Specifying More Frequent Global Contact Searches

By default for two-surface contact pairs, Abaqus/Explicit performs a more thorough search of the main faces near each secondary node every one hundred increments, which is sufficient for most analyses. However, there are some valid contact situations where a global search needs to be used more or less often during the step. *Figure 1* illustrates a situation that might require more frequent global tracking. The main surface is a valid surface, but it contains a hole. The secondary node shown identifies the shaded element facet as the closest main surface facet during an increment. The local contact search looks at this main surface facet and its neighbors.

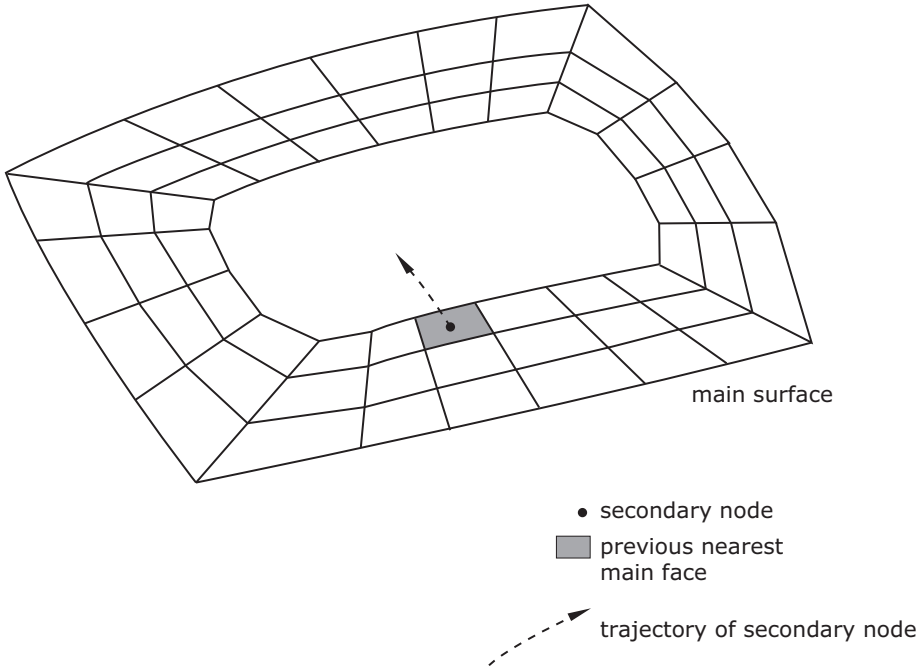


Figure 1: Example where local search may fail.

If the secondary node displaces across the hole in relatively few increments, the potential contact between the secondary node and the main surface facets across the hole will not be detected because the local contact search will still be checking the shaded facet. This same situation can occur when a secondary node moves rapidly across a deep valley in the main surface. The solution to this problem is to conduct global contact searches more frequently. You can specify the number of increments between global searches, n, for a given contact pair, if a value other than the default of 100 is desired.

Input File Usage: Use both of the following options:

**CONTACT PAIR*, CPSET=*contact_pair_set_name*
**CONTACT CONTROLS*, CPSET=*contact_pair_set_name*, GLOBTRKINC=*n*

Abaqus/CAE Usage: Interaction module:
Create Contact Controls: **Name:** *contact_controls_name*,
Abaqus/Explicit contact controls: **Specify max number of increments:** *n*

Interaction editor: **Contact controls:** *contact_controls_name*

Using a More Conservative Local Contact Search

The default local contact search used by Abaqus/Explicit uses techniques that allow it to use a minimum amount of computational time. If the local contact search has difficulty enforcing the appropriate contact conditions, a more conservative local contact search may resolve the problem. The contact search specified has no effect on contact pairs using self-contact.

Input File Usage: Use both of the following options:

**CONTACT PAIR*, CPSET=*contact_pair_set_name*
**CONTACT CONTROLS*, CPSET=*contact_pair_set_name*, FASTLOCALTRK=NO

Abaqus/CAE Usage: Interaction module:
Create Contact Controls: **Name:** *contact_controls_name*,
Abaqus/Explicit contact controls: toggle off **Fast local tracking**
Interaction editor: **Contact controls:** *contact_controls_name*

Tracking Contact with Highly Warped Surfaces

Calculating the correct contact conditions along a surface that is highly warped is very difficult, especially when the relative velocity of the contacting surfaces is very large. By default, Abaqus/Explicit monitors the orientation of every deformable main surface formed by element faces every 20 increments to check that the surface is not highly warped; rigid faceted surfaces are checked for large warping only at the beginning of a step. If a surface becomes highly warped, a warning message is issued in the status (`.sta`) file (see *Contact Diagnostics in an Abaqus/Explicit Analysis*), and a more accurate algorithm is used to calculate each secondary node's nearest point on the warped main surface. The alternate algorithm provides a more accurate solution but uses slightly more computational time.

Redefining the Criteria for a Highly Warped Surface

By default, Abaqus/Explicit considers a surface to be highly warped when the angle between surface normals at the nodes of a facet varies by more than 20°. The maximum variation of the surface normal over a facet is called the out-of-plane warping angle. You can change the default value of the out-of-plane warping angle cutoff from step to step for any contact pair in the model.

Input File Usage: **CONTACT CONTROLS*, CPSET=*contact_pair_set_name*, WARP CUT OFF=*angle*

Abaqus/CAE Usage: Interaction module:

Create Contact Controls: **Name:** *contact_controls_name*,
Abaqus/Explicit contact controls: **Angle criteria for highly warped facet (degrees):** *angle*

Interaction editor: **Contact controls:** *contact_controls_name*

Modifying How Frequently Abaqus/Explicit Checks for Warped Surfaces

You can specify the frequency, in increments, at which Abaqus/Explicit checks for warped surfaces for any contact pair in the model. The frequency can be changed from step to step. Checking for warped surfaces more frequently (the default is every 20 increments) will cause a slight increase in computational time for the analysis.

Input File Usage: **CONTACT CONTROLS*, CPSET=*contact_pair_set_name*, WARP CHECK PERIOD=*n*

Abaqus/CAE Usage: Interaction module:

Create Contact Controls: **Name:** *contact_controls_name*, **Abaqus/Explicit contact controls**: **Warp check increment:** *n*

Interaction editor: **Contact controls:** *contact_controls_name*

Contact Difficulties and Diagnostics

In this section:

- *Resolving Contact Difficulties in Abaqus/Standard*
- *Resolving Contact Difficulties in Abaqus/Explicit*

Resolving Contact Difficulties in Abaqus/Standard

In this section:

- *Contact Diagnostics in an Abaqus/Standard Analysis*
- *Common Difficulties Associated with Contact Modeling in Abaqus/Standard*

Contact Diagnostics in an Abaqus/Standard Analysis

Products: Abaqus/Standard Abaqus/CAE

References:

- *Output to the Data and Results Files*
- *About General Contact in Abaqus/Standard*
- *About Contact Pairs in Abaqus/Standard*
- *Contact Formulations in Abaqus/Standard*
- **CONTACT PRINT*
- **PREPRINT*
- **PRINT*
- *Viewing diagnostic output*

Overview

Diagnostics of an Abaqus/Standard analysis can be used to:

- check the initial contact conditions in a model; and
- track contact statuses over the course of the analysis.

Diagnostic information is available in several locations:

- The output database
- The job diagnostics tool in the Visualization module of Abaqus/CAE
- The data (`.dat`) file
- The message (`.msg`) file

Reviewing the Adjustments of Initially Overclosed Surfaces

Initial strain-free adjustments of nodal positions are performed by Abaqus/Standard under various circumstances to remove contact overclosures (see *Contact Initialization for General Contact in Abaqus/Standard* and *Contact Initialization for Contact Pairs in Abaqus/Standard*) or to remove overclosures or gaps between surfaces of surface-based tie constraints (see *Mesh Tie Constraints*). The initial configuration of the model is determined after these strain-free adjustments are applied. There are two sources of information on the adjustments of overclosed surfaces: the data (`.dat`) file and the output database (`.odb`) file.

Output of Information on Strain-Free Adjustments to the Data File

By default, information about a limited number of strain-free nodal adjustments is provided in the data (`.dat`) file. Requesting more detailed output concerning contact constraints provides information for all strain-free adjustments, regardless of the number of nodes adjusted.

Input File Usage: **PREPRINT*, CONTACT=YES

Abaqus/CAE Usage: Job module: job editor: **General**: **Preprocessor Printout**: **Print contact constraint data**

Visualizing Strain-Free Adjustments

Output variable STRAINFREE (see *Abaqus/Standard Output Variable Identifiers*) contains nodal vectors representing initial strain-free adjustments. By default, this output variable is written to the output database (`.odb`) file for the original field output frame at zero time if any strain-free adjustments are made by Abaqus/Standard. A symbol plot of this variable in the Visualization module of Abaqus/CAE shows vectors that represent how individual nodes have been adjusted, and a contour plot of this variable shows the distribution of the adjustment magnitude (you must select the original output frame at zero time in the Visualization module of Abaqus/CAE before choosing the STRAINFREE output variable). Initial nodal positions written to the output database file by Abaqus/Standard include the effects of strain-free adjustments, so plots of the initial configuration show the adjusted nodal positions.

Reviewing Initial Contact Conditions

Before conducting an analysis, perform a data check on the model to review the initial contact conditions (see *Abaqus/Standard and Abaqus/Explicit Execution*). The data check creates an output database and calculates the variable COPEN (contact opening) on each secondary surface based on the initial configuration of the model. You can create a contour plot of COPEN in the Visualization module of Abaqus/CAE to check for overclosed surfaces in the model assembly (an overclosure corresponds to a negative value of COPEN).

In addition, you can instruct Abaqus to print detailed information about the initial contact conditions to the data file during the data check (this information is not printed by default). The data file lists the status (open or closed) and clearance distance for each constraint point on a secondary surface, the internally generated contact element number associated with each secondary node or facet, and a summary of contact interaction properties. Internally generated contact elements are not user-defined and do not appear in the input file, so they can be difficult to locate if an error or warning message refers to them. The information in the data file can be used to locate these contact elements in the model.

The data file also lists the key parameters for every contact interaction in the model. These parameters include:

- main and secondary surface names;
- interaction property;
- value of h_{crit} (see *Controlling the Increment Size Based on Penetration Distance in Unconverged Iterations*);
- degree of smoothing on the main surface (see *Smoothing Main Surfaces for the Finite-Sliding, Node-to-Surface Formulation*);
- characteristic length used in penetration tolerance calculations (see *Augmented Lagrange Method*);
- extension ratio applied to main surface edges (see *Extending Main Surfaces and Slide Lines*); and
- contact formulation.

Parameters are listed only for the interactions to which they are applicable. For example, h_{crit}, surface smoothing, and the extension ratio are not used for surface-to-surface contact calculations (including general contact), so Abaqus does not report values for these parameters in surface-to-surface interactions.

Input File Usage: Use the following option to print information about initial contact conditions to the data file:

**PREPRINT*, CONTACT=YES

Abaqus/CAE Usage: Job module: job editor: **General**: **Preprocessor Printout: Print contact constraint data**

Output of Main Surface Nodes Associated with Secondary Nodes for Small-Sliding Contact

When you print initial contact conditions to the data file for contact pairs using the small-sliding tracking approach, Abaqus creates an output table showing the main nodes associated with each secondary node. Each row of the table lists a secondary node and the main nodes to which the secondary node transfers load when in contact with the main surface. The number of nodes in the table indicates whether or not the anchor point for a secondary node lies on an element face or at a node. For details on the small-sliding tracking approach and load transfer, see *Using the Small-Sliding Tracking Approach*.

In the output shown below for a two-dimensional model, secondary node 2 has an anchor point at main surface node 101 because it interacts with three main surface nodes. Secondary node 1 has an anchor point between nodes 100 and 101. This table also provides a list of secondary nodes that did not find an intersection with the main surface. This is important because these nodes have no local tangent plane and, hence, can penetrate the main surface.

```
SMALL SLIDING  NON-RIGID  AX ELEMENT(S)
        INTERNALLY GENERATED FOR SECONDARY BLANK AND MAIN SPHERE
         WITH SURFACE INTERACTION INF1

    ELEMENT  SECONDARY   MAIN
    NUMBER   NODE(S)     NODE(S)

       46       1     101     100
       47       2     102     101     100
       50       9          NO INTERSECTION
 ***WARNING: 1 SECONDARY NODES FOUND NO INTERSECTION WITH A MAIN
SURFACE
```

Tracking Contact Status during a Simulation

Abaqus provides two methods for tracking the status of contact interactions over the course of an analysis: the diagnostics tool available in the Visualization module of Abaqus/CAE and contact output to the data (`.dat`) file. Tracking contact status helps you ensure contact surfaces are defined appropriately, troubleshoot a terminated contact analysis, and verify that contact interactions behave realistically.

The diagnostics tool in Abaqus/CAE provides a good overview of how contact conditions evolve throughout a simulation. It is useful for reviewing terminated analyses because it reports contact change calculations in every iteration. The data file offers a more detailed summary of the overall contact conditions and the forces driving these conditions. However, it only provides output for successfully completed increments.

Contact Diagnostics in the Visualization module of Abaqus/CAE

The diagnostics tool in the Visualization module of Abaqus/CAE can be used with the following procedure types:

- static stress/displacement;
- coupled thermal/stress; and
- coupled pore fluid flow/stress.

The diagnostics tool tracks all changes in contact during an analysis. Each time a constraint point's contact status changes from closed to open, it is recorded as an “opening.” Each time the status changes from open to closed, it is recorded as an “overclosure.” If the contact interaction involves frictional effects, the diagnostics note when a constraint point begins sliding along the main surface (“slipping”) and when a constraint point in motion stops on the main surface (“sticking”). The diagnostics tool lists the constraint point involved in the status change and allows you to highlight the location of the constraint point in the model. The calculated clearance or overclosure

distance is also shown, and the maximum penetration is reported when the penetration tolerance for augmented Lagrange contact is exceeded (see *Augmented Lagrange Method*).

For the default contact convergence criteria, the diagnostics tool shows the maximum penetration error and the maximum estimated contact force error; these determine whether the contact conditions have converged (for details, see *Severe Discontinuities in Abaqus/Standard*). If you choose to use the traditional contact convergence criteria, these error measures are not reported. For analyses involving Lagrange friction, the diagnostics show the maximum slip error for points that should be sticking (see *Shear Stress Versus Elastic Slip While Sticking*).

For detailed instructions on using the diagnostics tool, see *Viewing diagnostic output*. The contact diagnostic information available in Abaqus/CAE can also be printed to the Abaqus message file. For details, see *The Abaqus/Standard Message File*.

Contact Output in the Data File

When you request contact output to the data file (see *Surface Output from Abaqus/Standard*), Abaqus lists the contact status for every constraint point at each increment of the analysis. The values of CPRESS, CSHEAR, COPEN, and CSLIP at each constraint point are also reported by default.

Example: Forming a Channel

Contact diagnostics are often helpful in confirming that the interactions in a model are behaving realistically and as intended. The diagnostics also provide a means of tracing the evolution of contact statuses on a node-by-node basis. In this example the diagnostics are based on a channel forming model. The channel is formed from a steel plate (or blank) with appreciable thickness. The blank is modeled with two-dimensional, plane strain elements; the forming tools (die, holder, and punch) are modeled as analytical rigid surfaces. The initial and final configurations of the model are displayed in *Figure 1*.

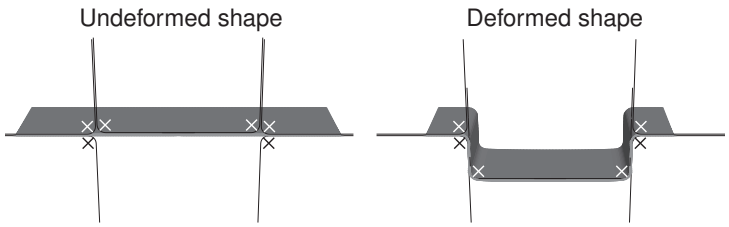


Figure 1: Model for channel-forming example. (The blank has been extruded for visualization purposes.)

If you include a step or prescribed condition in your model intended to establish contact between two surfaces, the diagnostics tool in Abaqus/CAE can confirm the success of this modeling technique. In this example contact must be firmly established between the blank, the die, and the holder before the forming process begins. Small but consistent overclosures in the nodes along the surface of the blank indicate that the contact conditions are appropriate to begin forming the channel (see *Figure 2*).

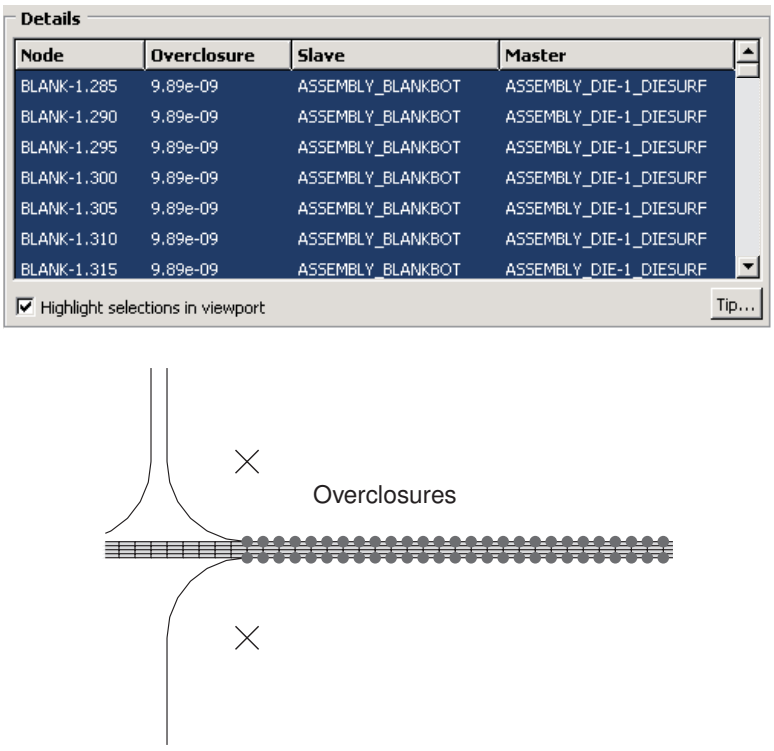


Figure 2: Diagnostics confirming contact conditions between the blank, die, and holder.

You can also use the contact conditions to review changes in contact status throughout the forming process. *Figure 3* depicts the onset of slipping for two nodes on the blank.

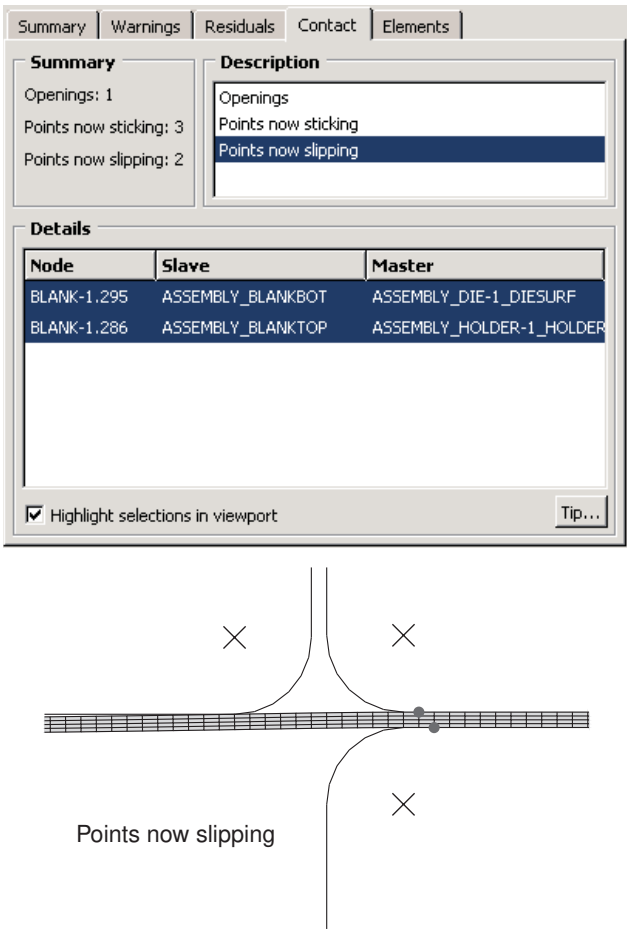


Figure 3: Diagnostics for the onset of slipping.

This information might be used to confirm frictional or material effects. For example, you can draw the following conclusions about these diagnostics in the channel forming analysis:

- If the slipping does not occur until well into the forming process, frictional forces were probably holding the blank in place between the die and holder.
- Since all the nodes on the blank do not slip simultaneously, there is most likely some mild stretching and nonuniform deformation occurring in the blank.

For more insight on the slipping nodes, refer to the data file. The following excerpt lists a portion of the blank-die interaction in the same increment depicted in *Figure 3*:

```
    NODE   FOOT-    CPRESS       CSHEAR1      COPEN        CSLIP1
           NOTE

      290   OP      0.000         0.000      4.1155E-07 -2.8783E-07
      295   SL    4.4632E+06 -4.4632E+05     0.000      -5.1137E-06
      300   ST    9.5643E+06 -9.3177E+05     0.000      -4.8711E-06
      305   ST    2.9421E+06 -2.7867E+05     0.000      -4.7359E-06
```

The contact status is indicated in the “footnote” column: open (OP), closed and sticking tangentially (ST), or closed and sliding tangentially (SL). In the absence of frictional properties the two contact statuses are open (OP) and closed (CL).

In the output above node 290 is open; consequently, the contact pressure variable CPRESS is zero. The COPEN variable reports that this node is 4.1155×10^{-7} length units away from the main surface. The SL footnote for node 295 indicates that it is in contact with the main surface (the die) and is "slipping." The critical shear stress, τ_{crit}, can be determined by the equation $\tau_{crit} = \mu p$, where p is the value of contact pressure shown under CPRESS and μ is the coefficient of friction for the contact interaction. In this model $\mu = 0.1$; the critical shear stress ($4.4632 \times 10^6 \times 0.1 = 4.4632 \times 10^5$) is equal to the frictional shear stress CSHEAR1, so the node is slipping. In the case of node 300 the critical shear stress ($9.5643 \times 10^6 \times 0.1 = 9.5643 \times 10^5$) is greater than the frictional shear stress, so the node is sticking. Likewise for node 305.

The CSLIP1 variable is the total accumulated (integrated) slip at the secondary node. Accumulated slip and local tangent directions are discussed in more detail in *Output of Tangential Results*.

Diagnosing a Terminated Contact Analysis

Contact diagnostics provide invaluable information when trying to resolve errors in a terminated analysis. The diagnostics let you review trends in the model's contact status, visually identify regions of the model involved in contact difficulties, and numerically quantify the severity of an error.

For a more general discussion of common errors associated with using contact in Abaqus/Standard analyses, refer to *Common Difficulties Associated with Contact Modeling in Abaqus/Standard*.

Excessive Severe Discontinuity Iterations

Establishing contact conditions is a common source of difficulty in an implicit static contact analysis. If an analysis terminates because it exceeds the maximum number of severe discontinuity iterations (see *Severe Discontinuities in Abaqus/Standard*), the contact diagnostics give insight into how to resolve the problem. You can plot the number of contact status changes over the course of an attempt, as shown in *Figure 4*.

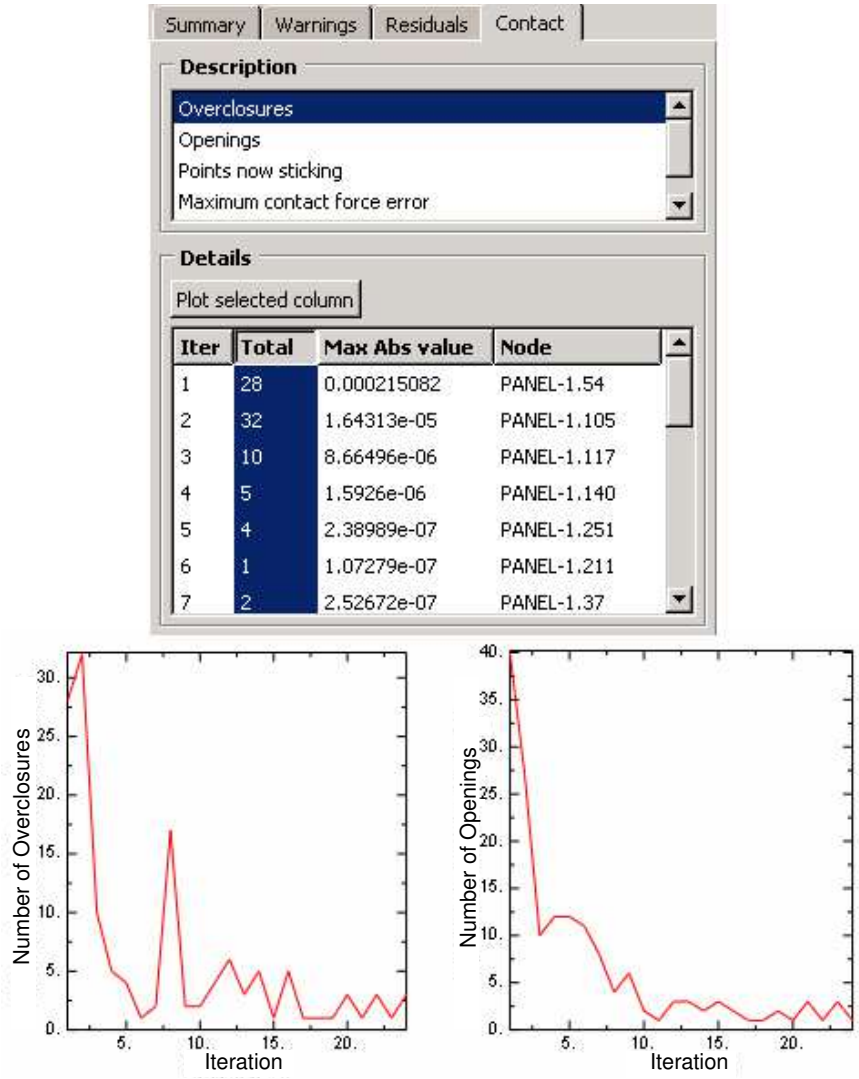


Figure 4: Changes in contact status during an attempt.

If the changes are tending toward zero, increasing the allowed number of severe discontinuity iterations or adjusting the SDI conversion settings may allow Abaqus to resolve the contact conditions. If the changes are not tending toward zero, you will need to revise your model or investigate other options.

Using the visualization tools, you can see which areas of the model are involved in contact changes. If a particular contact pair or surface region is causing a majority of the status fluctuations, you may need to modify the characteristics of the associated interaction. For example, it is typically easier to resolve contact conditions for contact pairs using the small-sliding tracking approach (if it is applicable) than for those using the finite-sliding tracking approach.

Chattering

The contact diagnostics tool makes it very easy to detect chattering in a model. In this situation the same node or constraint appears in the diagnostics summary for every iteration, alternating as an overclosure or an opening. The classic chattering scenario produces diagnostics plots that tend toward zero but level off at a low number due to the oscillating contact status (see *Figure 4*, for example). Techniques for resolving contact chattering problems are discussed in *Excessive Iterations in Contact Simulations*.

Unrealistic and Severe Overclosures

When reviewing diagnostics, you may notice overclosures during unconverged iterations for nodes or constraint points that are located outside of the regions that are contacting in a converged state. The reported overclosure value for these nodes will be significantly greater than the overclosures for nodes within the contacting regions, as seen in the highlighted constraint point in *Figure 5*.

Element	Face	Point	Overclosure	Slave	Master
PLATE-1.5	SPOS	1	0.215606	ASSEMBLY_PLATE-1	ASSEMBLY_PUNCH-1
PLATE-1.48	SPOS	3	139.554	ASSEMBLY_PLATE-1	ASSEMBLY_PUNCH-1
PLATE-1.269	SPOS	3	0.891122	ASSEMBLY_PLATE-1	ASSEMBLY_PUNCH-1
PLATE-1.269	SPOS	4	3.88193	ASSEMBLY_PLATE-1	ASSEMBLY_PUNCH-1
PLATE-1.274	SPOS	1	1.53256	ASSEMBLY_PLATE-1	ASSEMBLY_PUNCH-1

Figure 5: The overclosure at one constraint point is significantly higher than the overclosures at other constraint points.

This is an indication of physical or numerical instabilities in the model. You should take steps to more firmly establish contact before proceeding with the simulation or add some form of stabilization to the model (see *Solving Nonlinear Problems*, *Dashpots*, and *Automatic Stabilization of Rigid Body Motions in Contact Problems*). Using smaller increments can sometimes enable a solution to be obtained in these cases.

Nonconverging Force Equations

Contact diagnostics do not always involve severe discontinuity iterations. Poorly defined contact can lead to nonconvergence of the force equations in an analysis (see *Figure 6*).

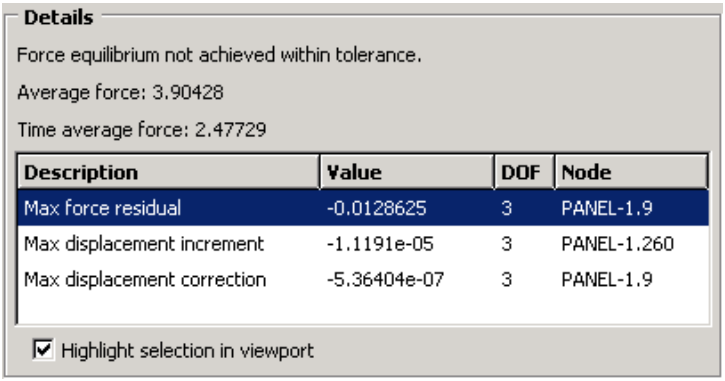


Figure 6: The diagnostics tool reports equilibrium difficulties.

If the same node appears repeatedly as the location of maximum residuals and corrections, investigate the contact conditions around that node. Consider the example in *Figure 7*.

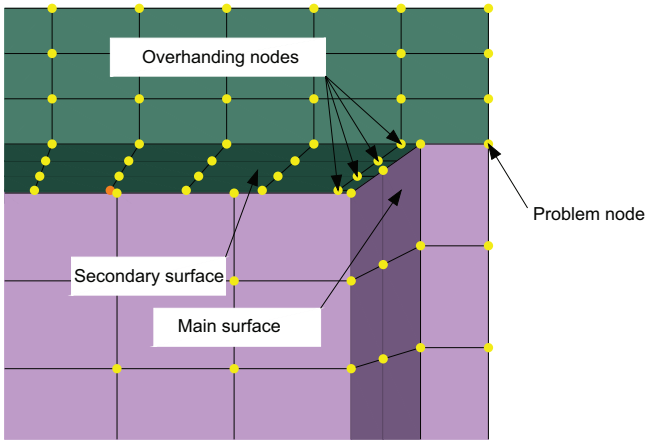


Figure 7: Two surfaces in a region of nonconverging force equations.

The diagnostics highlight the “problem node” on the perimeter of the secondary surface. A closer look in the vicinity of this node reveals that the secondary surface mesh is too coarse. Secondary nodes along the perimeter of the surface are touching the main surface, but the next row of nodes is “hanging over” the rim of the main surface. If this contact pair uses node-to-surface contact discretization, the main surface can penetrate the secondary surface with little resistance between the nodes. Such penetrations can cause the nonconverging force equations seen in the diagnostics.

Any situation in which the main surface is free to penetrate the secondary surface can prevent an analysis from converging. Potential solutions include:

- switching the main and secondary assignments;
- using surface-to-surface discretization (however, using surface-to-surface discretization without refining a coarse secondary mesh may lead to inaccurate stress results, even if the analysis does converge); or
- refining the mesh on the secondary surface.

Common Difficulties Associated with Contact Modeling in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *About General Contact in Abaqus/Standard*
- *About Contact Pairs in Abaqus/Standard*
- **CONTACT*
- **CONTACT PAIR*
- **CONTACT INITIALIZATION DATA*
- *Defining general contact*
- *Defining surface-to-surface contact*
- *Using contact and constraint detection*

Overview

This section highlights the difficulties that are most commonly encountered when modeling contact interactions with Abaqus/Standard. Recommendations on how to circumvent these problems are presented.

Difficulties Resolving Initial Contact Conditions

It is important to understand how Abaqus/Standard interprets and resolves contact conditions at the start of a step or analysis. If necessary, you can check initial contact conditions in the message file (see *The Abaqus/Standard Message File*). Unintentional contact openings or overclosures can lead to poor interpretations of surface geometry, unintentional motion in a model, and failure of an analysis to converge.

Removing Initial Contact Openings and Overclosures

When modeling the contact between two faceted surfaces, it is often possible for small gaps or penetrations to occur at individual nodes. This problem is particularly common when the two surfaces have dissimilar meshes. Abaqus/Standard uses two default methods for dealing with initial penetrations:

- In general contact small initial overclosures are automatically adjusted to remove the penetrations.
- In contact pairs initial overclosures are interpreted as interference fits and resolved accordingly (see *Resolving Large Interference Fits* below).

You can improve the accuracy of a contact simulation by having Abaqus/Standard adjust the position of the secondary surface to ensure that all secondary nodes that should initially be in contact with the main surface start out in contact without any penetration (see *Contact Initialization for General Contact in Abaqus/Standard* and *Contact Initialization for Contact Pairs in Abaqus/Standard*). When an intended initial clearance or overclosure is small compared to typical dimensions of the bodies in contact and a small-sliding contact pair is used, you can specify the clearance or overclosure precisely (see *Additional Contact Initialization Options for Small-Sliding Contact in Abaqus/Standard*).

The small-sliding contact tracking approach is more sensitive than the finite-sliding tracking approach to initial local gaps at the contact interface. In small-sliding contact each secondary node interacts with a contact plane defined from the finite element approximation of the main surface, as discussed in *Contact Formulations in Abaqus/Standard*. Abaqus/Standard can define these planes only when each secondary node can be projected

onto the main surface. Having these secondary nodes start the simulation contacting the main surface allows Abaqus/Standard to form the most accurate contact planes for the secondary nodes.

Large Unintended Initial Overclosures

The contact initialization algorithm may occasionally infer large initial overclosures where you do not intend initial overclosures to exist. For example, specifying incorrect surface normals can cause the contact initialization algorithm to interpret a physical gap as a penetration, as discussed in *Orientation Considerations for Shell-Like Surfaces*. Minor changes to the surface or contact definition will typically avoid undesired overclosures, but these situations typically call for some diagnosis to determine how to avoid the problem.

Identifying the Location of Unintended Overclosures

The first step in resolving a large initial overclosure is to identify the location of the problem:

- If initial overclosures are treated as interference fits to be resolved in the first increment (which is the default behavior for contact pairs; see *Modeling Contact Interference Fits in Abaqus/Standard*), a contour plot of the contact opening distance output variable (COPEN) for the initial output frame will show which regions have initial overclosures (penetrations correspond to negative values of COPEN).
- If initial overclosures are resolved with strain-free adjustments, a contour plot of the output variable STRAINFREE for the initial output frame will show where adjustments occurred (see *Contact Diagnostics in an Abaqus/Standard Analysis* for further discussion of this output variable). However, large strain-free adjustments may cause the mesh to become highly distorted, making it difficult to fully diagnose the problem; in such cases, perform a datacheck analysis (see *Abaqus/Standard and Abaqus/Explicit Execution*) with initial overclosures instead treated as interference fits to be resolved in the first increment to facilitate diagnosis (as discussed above).

Once you identify the location of an unintended initial overclosure, limiting the display in the Visualization module of Abaqus/CAE to the main and secondary surfaces of the interaction involved in the initial overclosure is helpful for identifying the cause of an unintended initial overclosure (see *Managing display groups* for a discussion of the display group options). Viewing the surface normals (see *Displaying element and surface normals*) may help determine whether unintended overclosures are due to incorrect surface normals.

Overclosures on Discontinuous Surfaces

Figure 1 shows an example with a large, unintended initial overclosure. In this case a single contact pair with discontinuous surfaces is meant to enforce contact in two distinct regions (*Table 1Orientation Considerations for Shell-Like Surfaces* shows which contact formulations allow discontinuous surfaces). The arrows in *Figure 1* show the positive normal direction for each surface region. The surface-to-surface contact formulation searches along the secondary-surface normal direction (in the positive and negative directions) for potential interaction points on the main surface. The search emanating from point A identifies point B as the only potential interaction point for point A in this example. The contact pair interprets this as a valid penetration because no better candidate interaction location is found and surface normals are opposed at points A and B. Methods to avoid this unintended overclosure include:

- defining separate contact pairs with continuous surfaces for each of the two distinct contact regions; and
- specifying general contact, which filters out nearly all unintended initial overclosures.

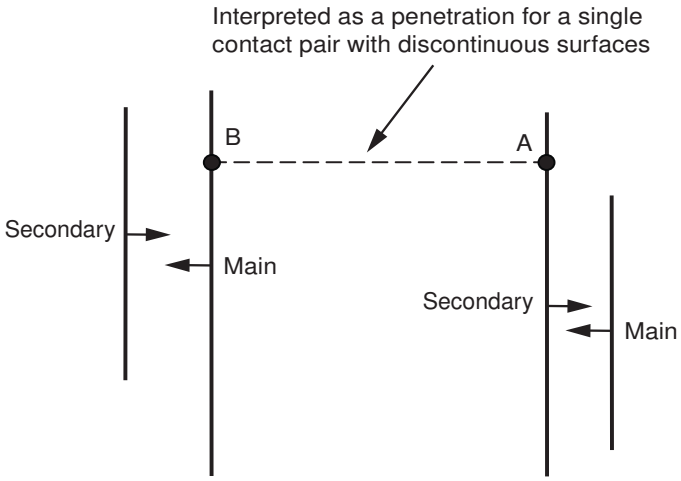


Figure 1: Example of an unintended initial overclosure due to a modeling error involving discontinuous surfaces.

Overclosures on Three-Dimensional Surfaces

The cause of unintended initial overclosures may be less obvious for three-dimensional models with complex surfaces. The most important step in overcoming this problem is identifying which regions of respective surfaces are involved in an unintended initial overclosure. For a surface-to-surface contact pair without strain-free adjustments, a portion of the main surface should be apparent behind the secondary surface (opposite the secondary surface normal direction) at a distance consistent with the reported (negative) COPEN value. For a node-to-surface contact pair, the direction to the interaction point on the main surface typically corresponds to a local minimum distance between the secondary and main surfaces.

Resolving Large Interference Fits

As previously discussed, Abaqus/Standard optionally interprets initial overclosures as interference fits. You should use one of the methods discussed above to remove any initial overclosures that are an unintended result of mesh discretization or errors in defining contact surfaces. In some cases the interference fit may be intended but may be too large to be resolved robustly with the method that is used by default for contact pairs in Abaqus/Standard (which is to resolve overclosures in a single increment). In this situation you should modify the contact model to allow resolution of overclosures over multiple increments (see *Modeling Contact Interference Fits in Abaqus/Standard* for more information). If you choose to have initial overclosures treated as interference fits for general contact, they are automatically resolved over multiple increments (see *Contact Initialization for General Contact in Abaqus/Standard*).

Preventing Rigid Body Motion in Contact Simulations

Rigid body motion is generally not a problem in dynamic analysis. In static problems rigid body motion occurs when a body is not sufficiently restrained. “Numerical singularity” warning messages and very large displacements indicate unconstrained motion in a static analysis. Therefore, if contact is used to constrain rigid body motion in static problems, ensure that the appropriate surface pairs are initially in contact (see *Contact Initialization for General Contact in Abaqus/Standard* and *Contact Initialization for Contact Pairs in Abaqus/Standard*). If necessary, define the model geometry to give a small initial overclosure to the contact pair, or use boundary conditions to move the structures into contact in the first step. The boundary conditions, which are unnecessary in subsequent steps, can be removed after the body is adequately constrained through contact with other components. Similarly, if a rigid body is meant to translate only, constrain its rotational degrees of freedom.

Frictional sticking can constrain rigid body motion. However, contact pressure must develop before friction can be generated. Therefore, friction is not effective in constraining rigid body motion when surfaces first come into

contact. You must temporarily eliminate rigid body motion by defining a boundary condition or by grounding the body with soft springs or dashpots.

If you are unable to prevent rigid body motion through modeling techniques, Abaqus/Standard offers some tools to automatically stabilize rigid bodies in contact simulations. These tools are discussed in *Automatic Stabilization of Rigid Body Motions in Contact Problems*.

Poorly Defined Surfaces

Over the course of an analysis, you may notice undesirable behavior between contact surfaces (excessive penetration, unexpected openings, inaccurate application of forces, etc.). This behavior often results in nonconvergence and termination of an analysis. These problems can arise from a number of causes related to mesh, element selection, and surface geometry.

Defining Duplicate Nodes on the Main Surface

When defining three-dimensional surfaces for use in finite-sliding applications, avoid defining two surface nodes with the same coordinates. Such a definition can give rise to a seam, or crack, in the surface as shown in *Figure 2*.

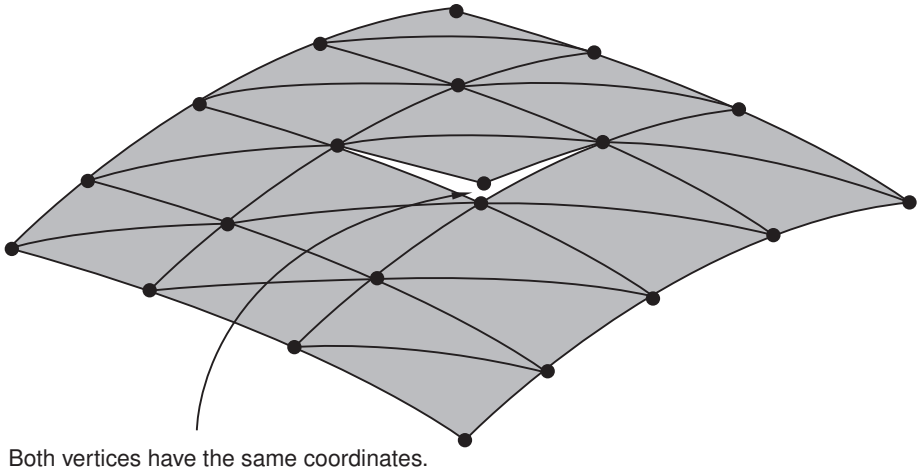


Figure 2: Example of doubly defined surface node.

If viewed with the default plotting options in Abaqus/CAE, this surface will appear to be a valid, continuous surface; however, if this surface is used as the main surface for finite-sliding, node-to-surface contact, a secondary node sliding along the surface may fall through this crack and get “stuck” behind the main surface. Similar problems can occur for finite-sliding, surface-to-surface contact. Typically, convergence problems will result that may cause Abaqus/Standard to terminate the analysis.

Use the edge display options in the Visualization module of Abaqus/CAE to identify any unwanted cracks in the surfaces used in the model. The cracks will appear as extra perimeter lines in the interior of the surface. Duplicate nodes can be avoided easily by equivalencing nodes when creating the model in a preprocessor.

Avoiding Problems with Contact along the Perimeters of Surfaces

When modeling finite-sliding contact, ensure that the main surface definition extends far enough to account for all expected motions of the contacting parts. Contact along the perimeter of main surfaces should be avoided with the node-to-surface contact formulation.. Abaqus/Standard assumes that the mating secondary surface nodes can fall off the free edge of the main surface, which can cause problems if a secondary node wraps around and

approaches its mating main surface from behind. *Figure 3* illustrates appropriate and inappropriate main surface definitions.

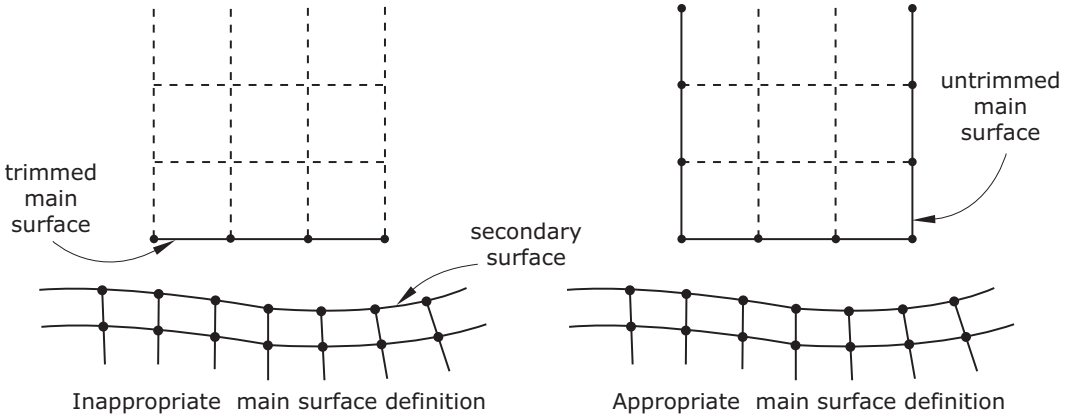


Figure 3: Example of main surface extension.

A secondary node that falls off a main surface in one iteration may find itself contacting the surface in the very next iteration; this phenomenon is known as chattering. If chattering continues, Abaqus/Standard may not be able to find a solution. This problem is less likely with the surface-to-surface formulation approach, because each contact constraint is based on a region of the secondary surface rather than individual secondary nodes. Request detailed contact printout to the message (`.msg`) file to monitor the history of a secondary node that might slide off the main surface (see *The Abaqus/Standard Message File*). The message file output will show the cyclic opening and closing of contact at a secondary node, which will indicate where the main surface needs to be modified.

For node-to-surface contact you can extend the main surface beyond the perimeter of the physical body that it approximates to avoid chattering problems. Chattering can also occur with some contact elements, such as slide line and rigid surface contact elements. Slide line contact elements can also be extended. See *Extending Main Surfaces and Slide Lines* for details.

Falling off Small-Sliding Main Surfaces

Falling off the edge of a main surface in small-sliding contact problems is not an issue since secondary nodes do not slide on the actual surface of the model. Instead, each secondary node interacts with a flat, infinite contact plane. This plane is associated with the set of main surface nodes that are closest to the secondary node in the undeformed configuration. For details about small-sliding contact, see *Contact Formulations in Abaqus/Standard*.

Falling off Surfaces Modeled with Interface Elements

Falling off the edge of a surface modeled with interface elements is not an issue since the secondary nodes slide on a flat, infinite contact plane.

Using Poorly Meshed Surfaces

Several problems are caused by surfaces created on very coarse meshes. Some of these problems depend on your choice of contact discretization, as discussed later in *Discrepancies between Contact Formulations*.

Penetrations with Coarsely Meshed Secondary Surfaces

When a coarsely meshed surface is used as a secondary surface for node-to-surface contact, the main surface nodes can grossly penetrate the secondary surface without resistance (see *Figure 4*). This situation is common when nonmatching meshes come into contact. Refining the secondary surface tends to alleviate this problem.

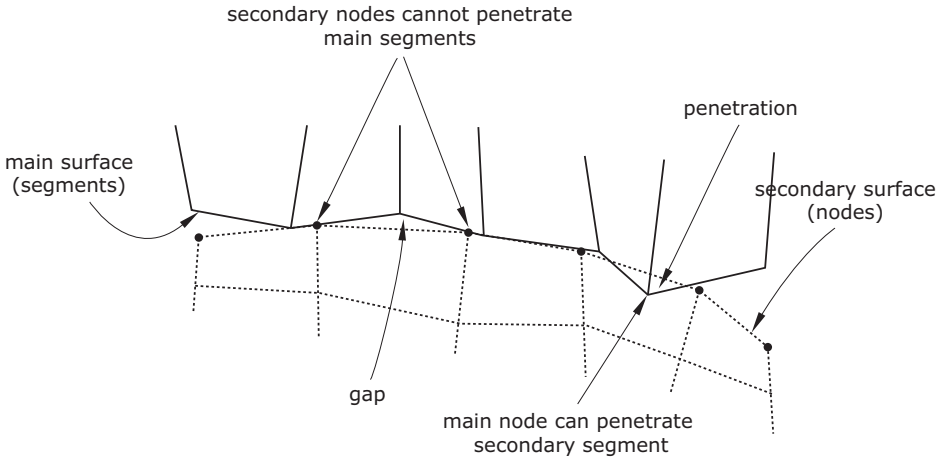


Figure 4: Main surface penetrations into the secondary surface due to a coarse mesh of the secondary surface for node-to-surface contact.

Surface-to-surface contact will generally resist penetrations of main nodes into a coarse secondary surface; however, this formulation can add significant computational expense if the secondary mesh is significantly coarser than the main mesh (see *Contact Formulations in Abaqus/Standard* for further discussion).

Contact Occurring at a Single Element

If the mesh on a surface is too coarse, it is possible for a contact interaction to occur entirely within the bounds of a single element. This typically happens when the two contacting surfaces have dissimilar curvature, as depicted in *Figure 5*.

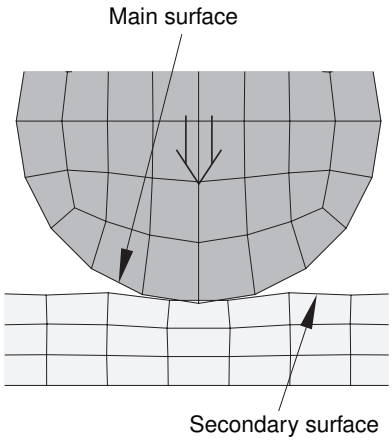


Figure 5: The main surface contacts the secondary surface at a single element face.

The results from such an interaction are unreliable and generally unrealistic. If the model in *Figure 5* uses node-to-surface contact, the main surface penetrates the secondary surface without resistance until it encounters a secondary node, as discussed above. If the main and secondary designations are reversed, the contact constraint is applied at a single secondary node; this concentration creates inaccurately high calculations of the contact pressure. If the model uses surface-to-surface contact, excessive penetration is not likely to occur. However, with only a small number of constraint points involved in the interaction, the averaging algorithm used to enforce surface-to-surface contact performs poorly. Inaccurate contact stress and pressure calculations result.

If contact is occurring at a single element, refine the mesh to spread the interaction across multiple element faces.

Coarsely Meshed Main Surfaces and Small-Sliding Contact

Coarsely meshed, curved main surfaces in small-sliding simulations can lead to unacceptable solution accuracy due to the approximate nature of the "main planes." Using a more refined mesh to define the main surface will improve the overall accuracy of the solution in small-sliding problems. However, unless perfectly matching meshes are used, local oscillations in the contact stress may still be observed, even in refined models.

Nonmatched Surface Meshes with Second-Order Heat Transfer Elements

Inaccurate local results may occur if second-order heat transfer elements are used to model a thermal interface and the meshes do not match across the surfaces. The worst results will be obtained when the midside node of an element on one surface is closest to the corner node of an element on the other surface. If a nonmatching mesh must be used in the model, use first-order elements or use a more refined mesh.

Three-Dimensional Surfaces with Second-Order Faces and a Node-to-Surface Formulation

Second-order elements not only provide higher accuracy but also capture stress concentrations more effectively and are better for modeling geometric features than first-order elements. Surfaces based on second-order element types work well with the surface-to-surface contact formulation but, in some cases, do not work well with the node-to-surface formulation (see *Contact Formulations in Abaqus/Standard* for a discussion of these contact formulations).

Some second-order element types are not well-suited for underlying the secondary surface with the combination of a node-to-surface contact formulation and strict enforcement of "hard" contact conditions, because of the distribution of equivalent nodal forces when a pressure acts on the face of the element. As shown in *Figure 6*, a constant pressure applied to the face of a second-order element without a midface node produces forces at the corner nodes acting in the opposite sense of the pressure.

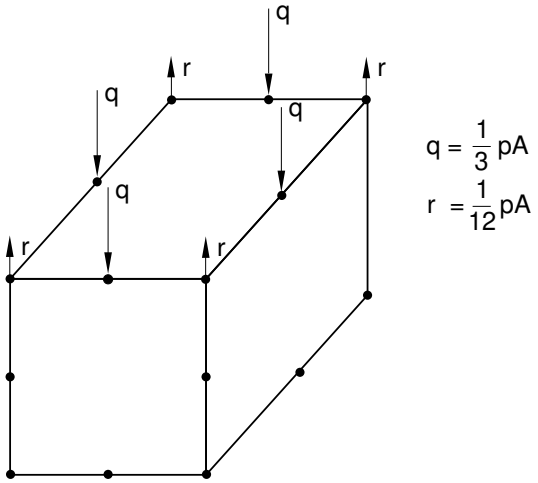


Figure 6: Equivalent nodal loads produced by a constant pressure on the second-order element face in "hard" contact simulations.

Abaqus/Standard bases important decisions for the node-to-surface contact formulation on contact forces acting on individual secondary nodes; the ambiguous nature of the nodal forces in second-order elements can cause Abaqus/Standard to make a wrong decision. To circumvent this problem, Abaqus/Standard automatically converts most three-dimensional second-order elements with no midface node (i.e., serendipity elements) that form a secondary surface into elements with a midface node. For the three-dimensional 18-node gasket elements, the midface nodes are also generated automatically if they are not given in the element connectivity. The presence of the midface node results in a distribution of nodal forces that is not ambiguous for the contact algorithm.

The element families C3D20(RH), C3D15(H), S8R5, and M3D8 are converted to the families C3D27(RH), C3D15V(H), S9R5, and M3D9, respectively. Since Abaqus/Standard does not convert second-order coupled temperature-displacement, coupled thermal-electrical-structural, and coupled pore pressure–displacement elements, you should specify a penalty or augmented Lagrange constraint enforcement method to approximate hard pressure-overclosure behavior (see *Contact Constraint Enforcement Methods in Abaqus/Standard*). Abaqus/Standard will interpolate nodal quantities, such as temperature and field variables, at the automatically generated midface nodes when values are prescribed at any of the user-defined nodes. Abaqus/Standard does not convert second-order serendipity elements if the secondary surface is used in a tied contact pair.

Second-order tetrahedral elements (C3D10 and C3D10HS) have zero contact force at their corner nodes. This combination of second-order triangular secondary facets, a node-to-surface contact formulation, and strict enforcement of "hard" contact conditions is disallowed to avoid a high likelihood of convergence problems and poor predictions of contact pressures that would occur with this combination. To avoid this combination, use at least one of the following alternatives:

- Use the surface-to-surface contact formulation (generally recommended) instead of the node-to-surface contact formulation;
- Use the penalty constraint enforcement method (generally recommended) or augmented Lagrange constraint enforcement method instead of strict enforcement of "hard" contact conditions; or
- Use modified 10-node tetrahedral elements (C3D10M) instead of second-order tetrahedral elements.

Excessive Iterations in Contact Simulations

Abaqus/Standard offers a number of methods to adjust the solver iteration scheme, sometimes resulting in a more efficient analysis with a minimal effect on accuracy.

Converting Severe Discontinuity Iterations in Weakly Determined Contact Conditions

By default, Abaqus/Standard continues to iterate until the severe discontinuities associated with changes in contact status are sufficiently small (or no severe discontinuities occur) and the equilibrium (flux) tolerances are satisfied. Alternatively, you can choose a different approach in which Abaqus/Standard continues to iterate until no severe discontinuities occur. These two approaches are discussed in more detail in *Severe Discontinuities in Abaqus/Standard*. The default treatment of severe discontinuity iterations reduces the likelihood of excessive iterations associated with chattering between contact states when the contact conditions are weakly determined. An example of a region with weakly determined contact conditions is near the center of a flat punch that contacts a thin plate supported at its edges.

Controlling the Increment Size Based on Penetration Distance in Unconverged Iterations

For most types of contact, if during an iteration the penetration calculated for any contact pair exceeds a specific distance (h_{crit}), Abaqus/Standard abandons the increment and tries again with a smaller increment size. There is no critical penetration distance for finite-sliding, surface-to-surface contact (including general contact) and for small-sliding contact in geometrically linear analyses.

The default value of h_{crit} is the radius of a sphere that circumscribes a characteristic surface element face. When calculating the default value, Abaqus/Standard uses only the secondary surface of the contact pair. The value of h_{crit} for each contact pair in the model is printed in the data (`.dat`) file. While the default value of h_{crit} should prove to be sufficient for the majority of contact simulations, in some cases it may be necessary to change the default value for a given contact pair. These cases include:

- Models in which the main surface is highly curved. The default value of h_{crit} may sometimes lead to situations as shown in *Figure 7*. During the iterative solution process a secondary node initially at point *a* may move to point *b*, penetrating the main surface with overclosure *h* less than h_{crit}. Abaqus/Standard may attempt to

move the secondary node to point *c* on the main surface. To avoid this situation, specify a smaller value for h_{crit} to force Abaqus/Standard to abandon the increment and to try a smaller increment size.

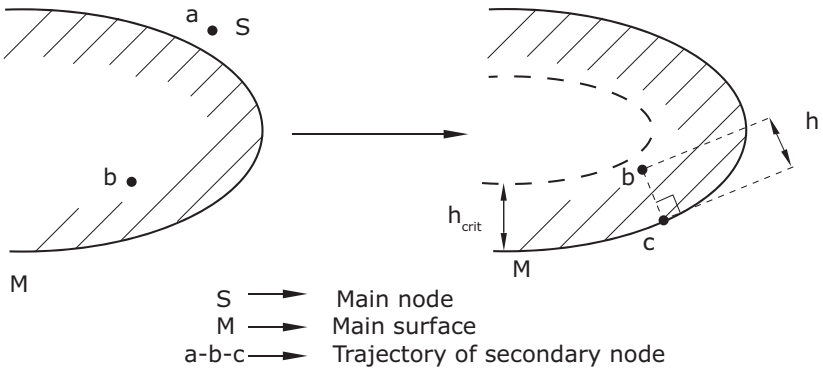


Figure 7: Effect of the critical penetration distance on a highly curved main surface.

- Models in which Abaqus/Standard cannot calculate a reasonable h_{crit} because a node-based surface is used. If there are other contact pairs in the model with surfaces, Abaqus/Standard uses the average dimension of all of the secondary surface element faces. If there are no other contact pairs, Abaqus/Standard uses a characteristic element dimension of the entire model.
- Models in which the contact face dimensions in a secondary surface vary greatly.
- Models in which the secondary surface mesh is very refined compared with the typical surface dimensions so that overclosures much larger than the default h_{crit} can be resolved easily.
- Models in which contact pairs with softened contact allow significant penetration (see *Contact Pressure-Overclosure Relationships*).

Input File Usage: **CONTACT PAIR*, HCRIT=h_{crit}

Abaqus/CAE Usage: You cannot adjust the default value of h_{crit} in Abaqus/CAE.

Difficulties Interpreting the Results of Contact Simulations

Although an analysis involving contact runs to completion, the results may seem unrealistic. This is sometimes due to modeling errors and sometimes due to the specialized output format of certain contact formulations. In addition to degrading contact output, the factors discussed below also tend to degrade convergence behavior, so avoiding these factors may improve convergence behavior.

Oscillating Contact Pressures When Using Second-Order Elements in "Hard" Contact Simulations

Nonuniform contact pressure distributions are likely to occur when very different mesh densities are used on the two deformable surfaces making up a contact interaction. The nonuniformity can be particularly pronounced when "hard" contact is modeled and both surfaces are modeled with second-order elements, including modified, second-order tetrahedral elements. In such cases oscillations and "spikes" in the contact pressure may occur. Smoother contact pressures may be obtained for surfaces modeled with second-order elements by using penalty-type contact constraint enforcement (see *Contact Constraint Enforcement Methods in Abaqus/Standard*).

Inaccurate Contact Stresses When Using Second-Order Axisymmetric Elements at the Symmetry Axis

For second-order axisymmetric elements the contact area is zero at a node lying on the symmetry axis $(r = 0)$. To avoid numerical singularity problems caused by a zero contact area, Abaqus/Standard calculates the contact area as if the node were a small distance from the symmetry axis. This may result in inaccurate local contact stresses calculated for nodes located on the symmetry axis.

Self-Contact

Contact of a surface with itself (self-contact) is provided for cases in which the original geometry is very different from the (deformed) geometry at which contact takes place. It would then be difficult for you to predict which parts of the surface will come into contact with each other. Where possible, it is always computationally more economical to declare parts of the surface as main and parts as secondary. The same unpredictability makes it impossible to determine a priori which side will be the main and which side the secondary. Therefore, Abaqus/Standard uses a symmetric contact model: every single node of the surface can be a secondary node and can simultaneously belong to main segments with respect to all other nodes.

Because each surface is acting as both a secondary and a main, the results of symmetric contact analyses can be confusing and inconsistent. These difficulties are discussed more fully in *Using Symmetric Main-Secondary Contact Pairs to Improve Contact Modeling*.

Overconstraining the Model

The term overconstraint refers to a situation in which multiple kinematic constraints outnumber the degrees of freedom on which they act. Overconstraints often lead to inaccurate solutions or failure to obtain a converged solution. Contact conditions strictly enforced with the direct constraint enforcement method (using Lagrange multipliers) are sometimes involved in overconstraints. See *Overconstraint Checks* for a detailed discussion and examples of overconstraints and how Abaqus/Standard will treat overconstraints based on the following classifications:

- Overconstraints detected in the model preprocessor
- Overconstraints detected and resolved during analysis
- Overconstraints detected in the equation solver

Abaqus/Standard will automatically resolve many types of overconstraints; however, many overconstraints involving contact cannot be resolved and will be exposed to the equation solver. The equation solver will often issue “zero pivot” or “numerical singularity” warning messages as a result of overconstraints; when this occurs, Abaqus/Standard will provide a warning message with information that is helpful for determining what contributed to the overconstraint so that you can resolve it. Occasionally overconstraints do not create warning messages; this does not necessarily mean that the overconstraints have not adversely affected the analysis.

Overconstraints Involving Softened Contact

Contact conditions with a softened behavior or enforced with the penalty or augmented Lagrange method will not combine with other constraints to cause “strict overconstraints”; however, “softened overconstraints” can:

- cause zero pivots or ill-conditioning in the equation solver if the stiffness contributions associated with contact are many orders of magnitude higher than the stiffness contributions from typical elements;
- prevent a tight penetration tolerance from being achieved with the augmented Lagrange method; and
- cause oscillations in contact stress solutions, particularly if the contact stiffness is high.

Some types of contact use the penalty or augmented Lagrange method by default to approximate hard pressure-overclosure behavior due to the prevalence of redundant or "competing" contact conditions. For a discussion of available constraint enforcement methods and default behavior, see *Contact Constraint Enforcement Methods in Abaqus/Standard*.

Inaccurate Contact Forces due to Overconstraints

If nodes in a contact pair are overconstrained but the equation solver does find a solution, the contact forces become indeterminate and may become excessively high, particularly in tied contact pairs. Check the time average force (or moment, or flux) reported in the message file, or use Abaqus/CAE to view the diagnostic information interactively (for more information, see *Viewing diagnostic output*). If it is many orders of magnitude larger than the residual forces (or moments, or fluxes), an overconstraint may have occurred, and there is no guarantee that Abaqus/Standard has found the correct solution. Another sign that the model is overconstrained is that the analysis begins to converge in a single iteration in every increment when the nonlinearities should require at least several iterations. Overconstraints should be avoided only by changing the contact definition or other constraint type involved.

Overconstraints due to Multiple Surface Interaction Definitions at a Single Node

Automatic resolution of contact overconstraints sometimes depends on whether two contact pairs refer to the same surface interaction definition. For example, consider a case in which two contact pairs have a common main surface and share some secondary nodes (perhaps along a common edge of two secondary surfaces). Overconstraints will occur at the common secondary nodes if the two contact pairs refer to different surface interaction definitions (even if the surface interactions are equivalent); however, Abaqus/Standard automatically avoids these overconstraints if the two contact pairs refer to the same surface interaction definition. (See *Assigning Contact Properties for Contact Pairs in Abaqus/Standard* for a discussion of how to assign surface interaction definitions to contact pairs.)

Discrepancies between Contact Formulations

The different contact formulations available in Abaqus/Standard (see *Contact Formulations in Abaqus/Standard*) allow for a great deal of flexibility when modeling contact simulations. However, two nearly identical simulations that differ only in the contact formulation being used will sometimes generate varying results. This is primarily because of the different ways that contact formulations interpret contact conditions. Certain formulations are better suited to particular situations.

Differences in Penetrations

The most observable difference between node-to-surface and surface-to-surface discretization is the amount of penetration that occurs between surfaces. This is because node-to-surface discretization computes penetrations only at secondary nodes, while surface-to-surface discretization computes penetrations in an average sense over a finite region. For example, when a secondary surface slides across a convex portion of a main surface, the secondary surface will tend to ride a bit higher with surface-to-surface discretization than with node-to-surface discretization, as shown in *Figure 8* (the opposite is true at a concave portion of a main surface). *Figure 9* shows another case in which the two contact discretizations behave fundamentally differently due to the different approaches to computing penetrations. Both discretizations converge to the same behavior as the mesh is refined.

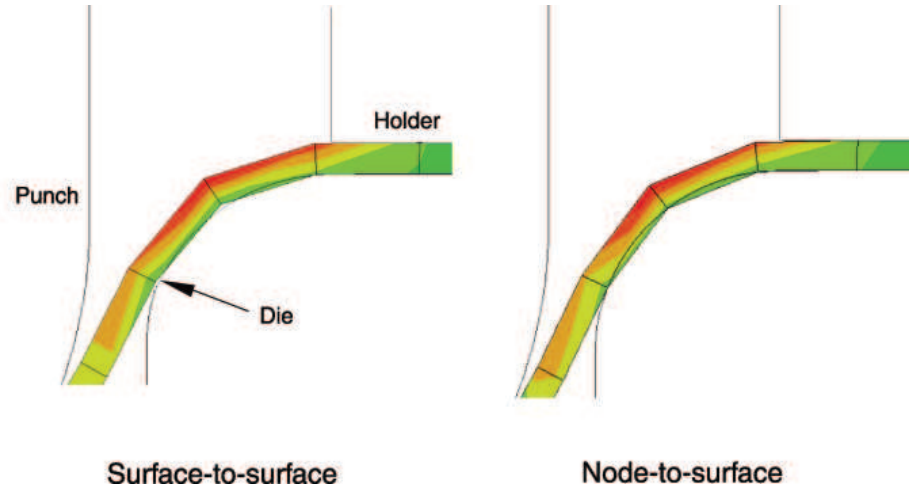


Figure 8: Comparison of contact discretizations in an example with convex curvature in the main surface (forming application).

The differences in computed penetrations can sometimes fundamentally affect the results of an analysis. Be aware of this possibility when converting models from one contact formulation to another. Various aspects of preexisting models, such as the friction coefficient or the pressure-overclosure relationship, may have been inadvertently "tuned" to the behavior that occurs with a particular contact formulation.

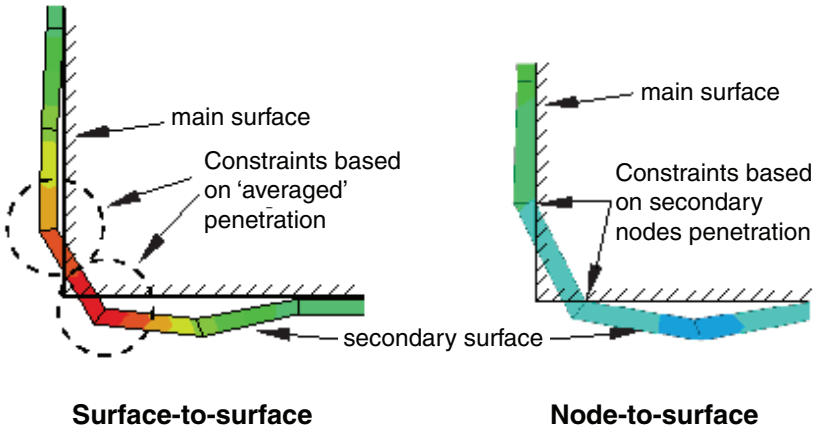


Figure 9: Comparison of contact discretizations in an example with a relatively flexible secondary surface wrapping around a corner of a main surface.

Contact at a Single Point

Figure 10 shows an example in which a circular rigid body is pushed into a deformable body.

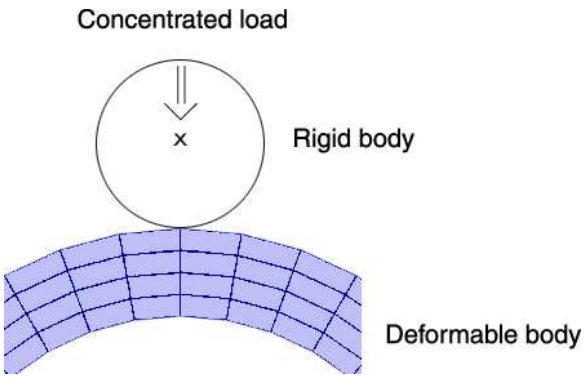


Figure 10: Example with two bodies initially touching at a single point.

In the initial configuration shown, the two bodies touch at a single point, which corresponds to a secondary node location. The following scenarios are likely for respective analyses of this model with node-to-surface and surface-to-surface discretization:

- With node-to-surface discretization, the first iteration is performed with one active contact constraint. A converged solution is obtained with a reasonable number of iterations and increments.
- With surface-to-surface discretization, penetrations are computed in an average sense over finite regions of the surface, so a positive gap distance is computed for all potential contact constraints even though the surfaces touch at one of the secondary nodes. However, the finite-sliding, surface-to-surface contact formulation detects that the surfaces are initially touching and by default automatically activates localized contact damping in the neighborhood where the gap distance is zero. Without such damping, Abaqus/Standard may not obtain a converged solution due to an unconstrained rigid body mode. This contact damping typically has an insignificant effect on the converged solution, and the damping is completely removed by the end of the step.

If you deactivate the automatic localized damping for the finite-sliding, surface-to-surface formulation—or if you are using the small-sliding, surface-to-surface formulation—you should use one of the techniques discussed above in *Difficulties Resolving Initial Contact Conditions* to remove the perceived initial gap between surfaces and prevent rigid body modes in the analysis.

Input File Usage: Use the following option to deactivate automatic localized contact damping at artificial surface gaps for contact pair definitions:

**CONTACT PAIR*, MINIMUM DISTANCE=NO

Use the following option to deactivate automatic localized contact damping at artificial surface gaps for general contact definitions:

**CONTACT INITIALIZATION DATA*, MINIMUM DISTANCE=NO

Abaqus/CAE Usage: You cannot deactivate automatic localized contact damping at artificial surface gaps in Abaqus/CAE.

Differences in Contact Normal Direction

Node-to-surface discretization uses a contact normal direction based on the main surface normal, whereas surface-to-surface discretization uses a contact normal direction based on the secondary surface normal (averaged over a region nearby the secondary node). For most active contact definitions the secondary and main surfaces are nearly parallel, so the main and secondary normals are approximately aligned; in which case this distinction

in how the contact normal is determined is not significant. However, in some cases the differences in the contact normal can be significant.

- When modeling large interference fits, surface-to-surface discretization can sometimes cause tangential motion of the secondary surface as the overclosures are resolved. This tangential motion may have undesirable effects on an analysis. See *Contact Initialization for General Contact in Abaqus/Standard* and *Modeling Contact Interference Fits in Abaqus/Standard* for more details.
- Contact constraints involving geometric edges of surfaces sometimes use a significantly different contact normal depending on which contact discretization approach is used, because the normals for the secondary and main surfaces may not directly oppose each other.
- The contact opening distance output variable (COPEN) can vary considerably depending on what type of contact formulation is used if the contact surfaces are not parallel. For node-to-surface discretization, the opening distance that is reported approximates the closest distance to the main surface; for surface-to-surface discretization, the opening distance that is reported corresponds to the distance from the secondary surface to the main surface along the secondary normal direction. The opening distance for surface-to-surface discretization is undefined if a line emanating from the secondary surface in the secondary normal direction does not intersect the main surface (as discussed in *Using the Small-Sliding Tracking Approach*, if a small-sliding constraint cannot be formed in such a case for the small-sliding, surface-to-surface formulation, Abaqus/Standard automatically reverts to the node-to-surface approach for individual constraints).

Contact at Corners

The finite-sliding, surface-to-surface formulation is often better-suited than other contact formulations for modeling contact near corners. In the example shown in *Figure 11*, the secondary surface is on the “outer” body (that is, the body with a reentrant corner). With node-to-surface discretization a single constraint acts at the corner secondary node in the “average” normal direction of the main surface, which often leads to poor resolution of contact, non-physical response, and even early termination of an analysis. However, surface-to-surface discretization generates two constraints near the corner for the respective faces, as shown in *Figure 11*, resulting in more stable contact behavior.

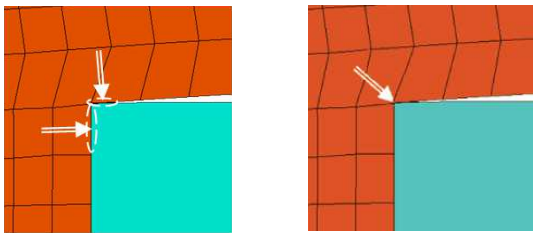


Figure 11: Comparison of contact formulations in an example with abutting surfaces having respective interior and exterior corners.

Resolving Contact Difficulties in Abaqus/Explicit

In this section:

- *Contact Diagnostics in an Abaqus/Explicit Analysis*
- *Common Difficulties Associated with Contact Modeling Using Contact Pairs in Abaqus/Explicit*

Contact Diagnostics in an Abaqus/Explicit Analysis

Products: Abaqus/Explicit Abaqus/CAE

References:

- *Output to the Data and Results Files*
- *About Contact Interactions*
- **DIAGNOSTICS*
- *Viewing diagnostic output*

Overview

Contact diagnostics in Abaqus/Explicit allow you to get detailed information about the surfaces and progress of contact interactions. Diagnostics are available:

- to review automatic adjustments between two surfaces,
- to reveal potentially problematic initial surface configurations in a model,
- to track excessive penetrations between two contacting surfaces, and
- to review warnings associated with contact between warped surfaces.

Reviewing the Adjustments of Initially Overclosed Surfaces

Contacting surfaces that are overclosed in the initial configuration of the model are adjusted automatically by Abaqus/Explicit to remove the overclosures (see *Contact Initialization for General Contact in Abaqus/Explicit* and *Contact Initialization for Contact Pairs in Abaqus/Explicit*). There are three sources of information on the adjustments of overclosed surfaces: the status (`.sta`) file, the message (`.msg`) file, and the output database (`.odb`) file.

Obtaining the Adjustments of Overclosed Surfaces in the Status and Message Files

By default, Abaqus/Explicit writes all nodal adjustments and—for general contact surfaces—contact offsets to the message (`.msg`) file along with a summary listing of the maximum initial overclosure and the maximum nodal adjustment to the status (`.sta`) file for the contact pairs defined in the first step of a simulation. You can choose to suppress the information written to the message file and write only the summary information to the status file. The information written to the message and status files is also written to the output database (`.odb`) for use in Abaqus/CAE.

Input File Usage: Use the following option to obtain both detailed diagnostic output to the message file and summary diagnostic output to the status file:

**DIAGNOSTICS*, CONTACT INITIAL OVERCLOSURE=DETAIL (default)

Use the following option to obtain only summary diagnostic output to the status file (no contact diagnostics will be written to the message file):

**DIAGNOSTICS*, CONTACT INITIAL OVERCLOSURE=SUMMARY

Abaqus/CAE Usage: You cannot control the diagnostic information for contact initial overclosures from within Abaqus/CAE. Use the following option to view the saved diagnostic information:

Visualization module: **Tools->Job Diagnostics**

Viewing the Adjustments of Surfaces

In the first step the adjustments of initially overclosed surfaces can be viewed in Abaqus/CAE. Displaced shape plots that show the adjustments to the contact pairs defined in the first step can be plotted for the original field output frame at zero time (this nodal coordinate adjustment may introduce small nonzero strain output for solid elements even when the stresses are zero). In addition, output variable STRAINFREE (see *Abaqus/Explicit Output Variable Identifiers*) contains nodal vectors representing initial strain-free adjustments. By default, STRAINFREE is written to the output database (`.odb`) file for the original field output frame at zero time if any strain-free adjustments are made by Abaqus/Explicit. A symbol plot of this variable in the Visualization module of Abaqus/CAE shows vectors that represent how individual nodes have been adjusted, and a contour plot of this variable shows the distribution of the adjustment magnitude (you must select the original output frame at zero time in the Visualization module of Abaqus/CAE before choosing the STRAINFREE output variable). In the case of overclosures in steps other than the first, vector plots of nodal displacements and accelerations can be particularly helpful in visualizing the adjustments. Such plots can be viewed in Abaqus/CAE after a data check analysis (see *Abaqus/Standard and Abaqus/Explicit Execution*).

Visualizing the Precise Initial Clearances for Small-Sliding Contact Pairs

Abaqus/Explicit does not adjust the coordinates of the secondary surface when precise initial clearances are specified for small-sliding contact pairs (see *Contact Initialization for Contact Pairs in Abaqus/Explicit*). Therefore, the specified clearances cannot be seen in a postprocessor such as the Visualization module of Abaqus/CAE. Thus, depending on the initial geometry of the surfaces and the magnitude of the clearances or overclosures, the surfaces may appear open or closed in the postprocessor when they are actually just in contact in the simulation.

Detecting Crossed Surfaces in a General Contact Domain

If a secondary surface initially penetrates a double-sided main surface by a distance greater than the main surface's thickness, the severely overclosed secondary nodes will see the back side of the main surface as the appropriate contact force direction. These secondary nodes in these crossed surfaces effectively become trapped behind the main surface. This issue is discussed in more detail in *Contact Initialization for General Contact in Abaqus/Explicit* and *Contact Initialization for Contact Pairs in Abaqus/Explicit*.

For general contact definitions, diagnostic testing that identifies regions in which surfaces are crossed in the initial configuration is activated by default. When the diagnostic tests are activated, a warning message is issued to the message (`.msg`) file if two adjacent secondary nodes (connected by a facet edge) are detected on opposite sides of a main surface. No such warning is issued for node-based surface nodes on opposite sides of a main surface, because adjacency cannot be determined among the node-based surface nodes. In some cases involving corners of main surfaces this warning message may be issued even though adjacent secondary nodes are really on the same side of a main surface. The CPU cost of performing diagnostic testing on large models is potentially significant. You can choose to deactivate the diagnostic testing and avoid the extra CPU cost in such cases.

Input File Usage: Use the following option to deactivate diagnostic testing for initially crossed surfaces:

**DIAGNOSTICS*, DETECT CROSSED SURFACES=OFF

Abaqus/CAE Usage: You cannot exclude diagnostic testing for initially crossed surfaces from within Abaqus/CAE. Use the following option to view the saved diagnostic information:

Visualization module: **Tools->Job Diagnostics**

Excessive Penetrations between General Contact Surfaces

As described in *Contact Constraint Enforcement Methods in Abaqus/Explicit*, the penalty constraint enforcement method used by the general contact algorithm in Abaqus/Explicit allows slight penetrations of one surface into another surface. A "spring" stiffness is applied automatically to the surfaces to resist these penetrations. If the nodes involved in general contact do not have adequate mass, the default "spring" stiffness chosen automatically by Abaqus/Explicit may not be sufficient to prevent large penetrations. Such a situation can arise, for example, when a cloud of massless nodes, fully constrained by a kinematic coupling definition, contacts a fully constrained rigid face with no mass.

By default, if during node-to-face contact, the penetration of a node into its tracked face exceeds 50% of the typical face dimension in the general contact domain, the penetration is regarded as excessive and Abaqus/Explicit issues a diagnostic message to the status (`.sta`) file. A node set containing deeply penetrated nodes is also written to the output database (`.odb`) file for use in Abaqus/CAE. You can control the fraction of the typical face dimension used to trigger the diagnostic message.

Input File Usage: Use the following option to control the fraction of the typical element face dimension used to trigger the diagnostic message for deep penetrations:

**DIAGNOSTICS*, DEEP PENETRATION FACTOR=*value*

Abaqus/CAE Usage: You cannot control the diagnostic information for deep penetrations from within Abaqus/CAE. Use the following option to view the saved diagnostic information:

Visualization module: **Tools->Job Diagnostics**

Warning Messages for Highly Warped Surfaces

Calculating the correct contact conditions along a surface that is highly warped is very difficult, and Abaqus/Explicit employs a specialized algorithm to enforce contact between warped surfaces; this specialized algorithm is more expensive than the default contact algorithm (see *Contact Controls for Contact Pairs in Abaqus/Explicit*). By default, Abaqus/Explicit checks for highly warped surfaces every 20 increments.

Abaqus/Explicit writes a warning message in the status (`.sta`) file the first time that it detects that a surface is highly warped. The message is brief; it states only which surface has a highly warped facet. If additional facets on this surface become highly warped later in the analysis, no additional warning messages are issued.

You can request more detailed diagnostic warning messages, if desired. In this case the message file will contain a warning every time a warped facet is found on a particular surface. The warnings will give the parent element associated with the warped facet (the parent element is the element whose face forms the facet) and the warping angle of the facet.

The computation time and the size of the message file can increase significantly if detailed warnings are requested. You can switch back to the summary warnings in subsequent steps or suppress the warped surface warnings entirely.

If the analysis terminates with a fatal error, the preselected output variables will be added automatically to the output database as field data for the last increment.

Input File Usage: Use the following option to request detailed diagnostic warning output for warped surfaces:

**DIAGNOSTICS*, WARPED SURFACE=DETAIL

Use the following option to request the default summary diagnostic output for warped surfaces:

**DIAGNOSTICS*, WARPED SURFACE=SUMMARY

Use the following option to suppress diagnostic warning output for warped surfaces entirely:

**DIAGNOSTICS*, WARPED SURFACE=OFF

Abaqus/CAE Usage: Diagnostic output requests for warped surfaces are not supported in Abaqus/CAE.

Common Difficulties Associated with Contact Modeling Using Contact Pairs in Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References:

- *About Contact Pairs in Abaqus/Explicit*
- **CONSTRAINT CONTROLS*
- **CONTACT PAIR*

Overview

This section highlights the difficulties that are most commonly encountered when modeling contact interactions with contact pairs in Abaqus/Explicit. Most of these issues are not relevant when the general contact algorithm is used; refer to *About General Contact in Abaqus/Explicit* for more information on the issues involved with general contact interactions. Recommendations on how to circumvent these problems are presented.

Defining Duplicate Nodes on the Main Surface

When defining three-dimensional surfaces formed by element faces, avoid defining two surface nodes with the same coordinates. Such a definition can give rise to a seam, or crack, in the surface as shown in *Figure 1*.

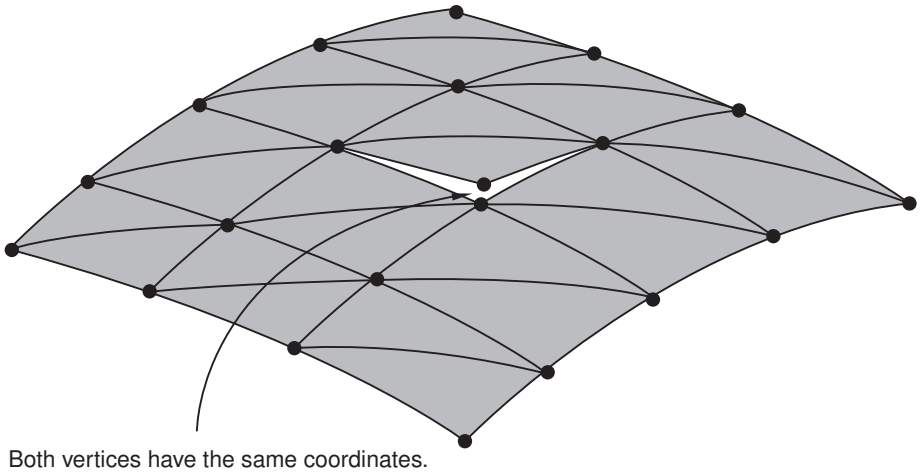


Figure 1: Example of doubly defined surface node.

If viewed with the default plotting options in Abaqus/CAE, this surface will appear to be a valid, continuous surface; however, a node sliding along this surface can fall through this crack and violate the contact conditions. If this were to happen, Abaqus/Explicit would enforce the contact conditions by applying a large acceleration to the node once overclosure is detected. The large resulting acceleration may create a noisy solution or cause the elements to distort badly.

Use the edge display options in the Visualization module of Abaqus/CAE to identify any unwanted cracks in the surfaces used in the model. The cracks will appear as extra perimeter lines in the interior of the surface. Duplicate nodes can be avoided easily by equivalencing nodes when creating the model in a preprocessor.

Using an Inadequate Surface Definition for the Desired Contact Conditions

Occasionally, surface definitions may not be suitable for modeling the desired contact conditions in a problem. *Figure 2* shows a two-dimensional model of a simple connection between two parts.

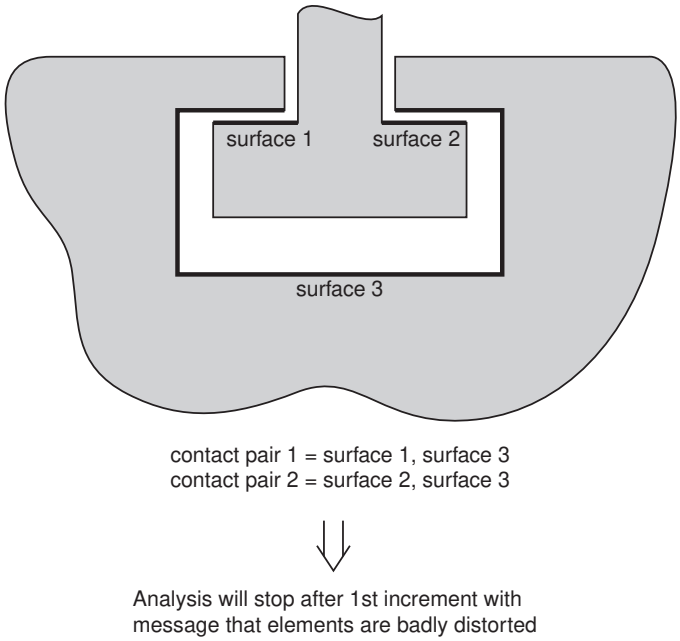


Figure 2: Surface definitions that are inadequate for the desired contact conditions.

The surfaces shown in the figure are inadequate for the desired contact conditions that are also shown. At the start of the simulation, Abaqus/Explicit will detect that some of the nodes on surface 3 are behind surfaces 1 and 2. When the contact conditions are enforced, the motions of the surfaces will likely cause badly distorted elements. One solution to this problem is shown in *Figure 3*.

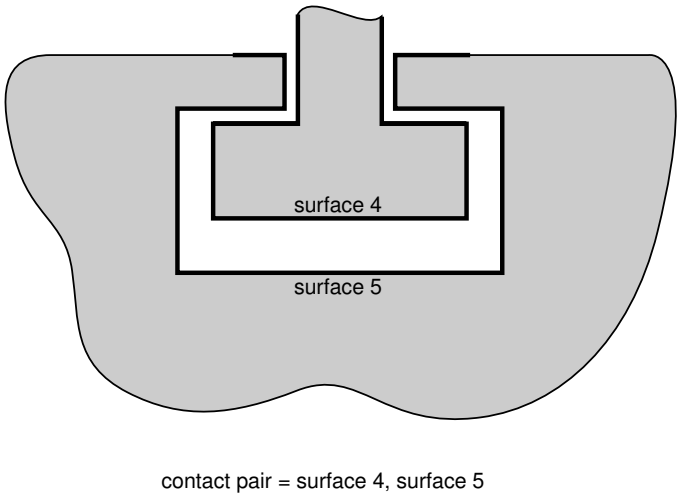


Figure 3: Surface definitions that are adequate for the desired contact conditions.

The surfaces shown in that figure are suitable for the desired contact definition. Other solutions, such as using a pure main-secondary contact pair, exist for this problem and may be more suitable, depending on the details of the intended simulation.

Using Poorly Discretized Surfaces

Several problems are caused by surfaces created on very coarse meshes.

Penetrations with Coarsely Discretized Surfaces When Using Hard Surface Behavior

When a coarsely discretized surface is used as the secondary surface in a pure main-secondary contact pair with hard surface behavior, an inaccurate solution may be produced as a result of the gross penetration of the main surface into the secondary surface. This situation is shown in *Figure 4*. This problem can be minimized if the contact pair can be switched to a balanced main-secondary contact pair. However, some contact pairs in Abaqus/Explicit must always use a pure main-secondary formulation. In these cases the only solution to gross penetration is to refine the secondary surface.

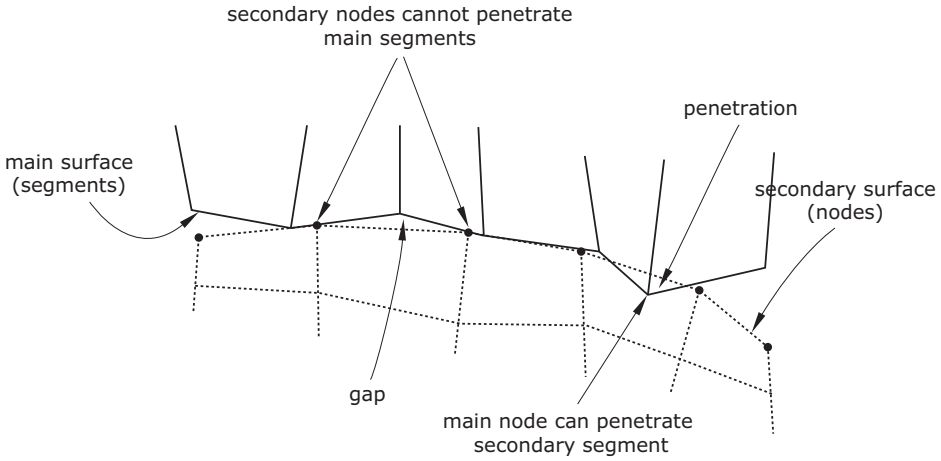


Figure 4: Main surface penetrations into the secondary surface due to coarse discretization.

Problems with Coarsely Discretized Rigid Surfaces

For rigid surfaces formed by element faces, inaccurate results may be obtained if too few elements are used to represent a curved geometry. When a very coarse mesh is used on a curved geometry, it is possible for secondary nodes to get “snagged” on the sharp vertices.

In general, using a reasonable number of element faces to represent a curved surface will not increase the computational time of the simulations. However, a large number of element faces can significantly increase the memory that Abaqus/Explicit will need for the simulation. When a specific curved surface geometry can be modeled, using an analytical rigid surface may provide a more accurate geometric description while minimizing computational expense; see *Analytical Rigid Surface Definition*.

Penalty Contact Behavior Sensitivity in Rigid-to-Rigid Interactions

The contact penalties are, in general, determined from stable time increment considerations and masses of the nodes involved in contact. To compute a reliable contact penalty when rigid bodies are contacting each other, Abaqus/Explicit accounts in a comprehensive fashion for the inertial properties of the rigid bodies by distributing the mass of the rigid bodies at all nodes that might be involved in contact. Hence, the final contact penalty will depend on the size of the actual rigid surfaces that are included in the contact definitions. Consequently, the contact response (forces, penetrations) will depend somewhat on your choice in defining the contacting surfaces on the rigid bodies. If large penetrations occur, specifying realistic inertial properties for the rigid bodies will help in general to resolve the issue. Alternatively, you can use a scaling factor for the penalties to enforce contact in a more accurate fashion.

Conflicts with Boundary Conditions

If boundary constraints are applied to contact nodes on both surfaces of a contact pair in the direction that the contact constraints are active, the boundary constraints may override the contact constraints. For kinematic contact, contact force related quantities will be output as the force necessary to resolve the contact constraint in a single increment, causing misleading results for these output quantities if the boundary constraints violate the contact constraints. Contact force output for penalty contact does not show this behavior since the contact force is proportional only to the current penetration and does not depend on the time increment. Boundary constraints are not affected by contact constraints.

Conflicts with Multi-Point Constraints

Using a multi-point constraint (MPC) with a node on a surface that is part of an active kinematic contact pair can generate conflicting kinematic constraints in the model. Abaqus/Explicit will not prevent you from using multi-point constraints on the nodes forming a surface. If the contact constraints and the constraints formed by the MPC are orthogonal, there will be no problems with the simulations. If they are not orthogonal, the solution may be noisy as Abaqus/Explicit tries to satisfy the conflicting constraints. Since within each increment kinematic contact constraints are applied after MPCs are applied, the MPCs on kinematic contact surfaces may be slightly out of compliance.

In the case of an interaction between an MPC and penalty contact, the MPC is strictly enforced and any noncompliance in the contact pair will be resisted by penalty forces.

Conflicting Contact Constraints on Shell Nodes with Hard Contact

When a shell or membrane is pinched between two main surfaces using two kinematic contact pairs with hard contact behavior, one of the contact constraints will not be enforced exactly. In a quasi-static analysis it may be observed that the pinched secondary node will oscillate about an “equilibrium” penetration depth with a decay rate that depends on the time increment and the ratio of the mass of the pinched node and the mass of the main surfaces. Decreasing the time increment size will increase the decay rate (quasi-static equilibrium will be reached more quickly). Reducing the mass of the nodes on the main surfaces (or increasing the mass of the pinched nodes) will also increase the decay rate, although a high ratio of secondary mass to main mass can also lead to numerical difficulties for kinematic contact, as discussed below in *Large Mass Mismatch between Contact Surfaces*. Applying the loads to the model gradually will reduce the amplitude of the oscillation. In most analyses it is not desirable to alter the time increment or nodal masses arbitrarily, so the decay rate of the oscillation will be fixed. Either the loading rate can be modified or a softened contact model with contact damping can be used to control this oscillatory behavior.

The quasi-static equilibrium penetration magnitude, p_{equil}, is approximately given by

$$p_{equil} = \frac{f\Delta t^2}{m},$$

where f is the normal contact force, Δt is the increment size, and m is the mass of the pinched node. The quasi-static equilibrium penetration will be minimal if it is small compared to the shell or membrane thickness. A change in the time increment size or loading on the pinched surfaces during the analysis causes the quasi-static equilibrium penetration to change, which can be responsible for large accelerations of surface nodes and can contribute to solution noise (typically, this behavior manifests as a jump in contact results such as CPRESS). Similar noisy behavior for pinched surfaces can occur across a step boundary, even if the time increment size is uniform across the step boundary.

If one kinematic contact pair and one penalty contact pair are used to model the same type of pinching problem, the kinematic constraint is enforced exactly and the static value of the penetration in the penalty contact pair is

somewhat larger than that which occurs when kinematic contact is used for both contact pairs (assuming that the penalty stiffness is set such that the analysis is numerically stable for the time increment being used).

Multiple Kinematic Contact Constraints on Solid Nodes

If a node that is not attached to shell or membrane elements acts as a secondary node in two or more simultaneous, kinematic contact constraints, the resulting contact corrections may be erroneous, possibly causing the analysis to end with excessive element distortion. By "not attached to shell or membrane elements" we are referring to nodes attached to solid elements or point masses, for example. The majority of solid nodes typically are not involved in simultaneous contacts, but there are common exceptions where three or more bodies meet at corners. This limitation can be avoided by using penalty contact. For example, if a solid surface acts as a secondary in two contact pairs and there is a possibility of simultaneous contacts for individual secondary nodes, penalty enforcement of contact should be specified for one or both of the contact pairs.

Redundant and Degenerate Contact Constraints

Redundant contact constraints are caused by overlapping or adjoining surfaces. For example, if contact is specified between a single surface and multiple overlapping surfaces, the contact constraints associated with the common nodes of the overlapping surfaces are redundant. Degenerate contact constraints occur if the secondary surface and main surface of the same contact pair contain common nodes (a contact constraint cannot be formed between a node and itself).

If redundant kinematic contact constraints are specified, Abaqus/Explicit will consolidate the constraints if both contact pairs use pure main-secondary contact, the secondary surfaces do not share facets, and the surface interaction and contact pair set names are identical. If the contact pair definitions differ, the analysis will terminate with an error, and one of the redundant constraints must be removed from the model definition to continue the analysis.

Redundant penalty contact constraints may cause excessive initial overclosure adjustments, creating gaps in the place of initial overclosures. To correct this behavior, one of the constraints must be removed from the model definition.

Redundant contact constraints involving both a penalty contact pair and a kinematic contact pair cause inefficiencies in the analysis. The kinematic contact constraints will override the penalty contact constraints, but the penalty contact constraints will still be considered in the automatic time increment estimate.

If the surfaces in a two-surface contact pair contain common nodes, the contact constraint for each shared node cannot be generated. This is the equivalent of defining self-contact between the shared nodes and each surface. However, the two-surface contact logic (unlike the specialized self-contact logic) would erroneously detect contact between each shared node and itself. When this condition occurs, Abaqus/Explicit redefines the secondary surfaces so that the shared nodes will not act as secondary nodes in the contact pair. However, the shared nodes will still be used in the definition of a main surface in the contact pair.

Large Mass Mismatch between Contact Surfaces

Often very little mass is assigned to rigid bodies in quasi-static simulations because the mass has little influence on the physical problem. However, specifying a small rigid body mass can adversely affect the kinematic contact enforcement method. A force applied to a rigid body with very little mass can cause a large predicted displacement of the rigid body within an increment prior to the enforcement of contact constraints, so significant penetration may be present in the "predicted" configuration for kinematic contact, as shown in *Figure 5*.

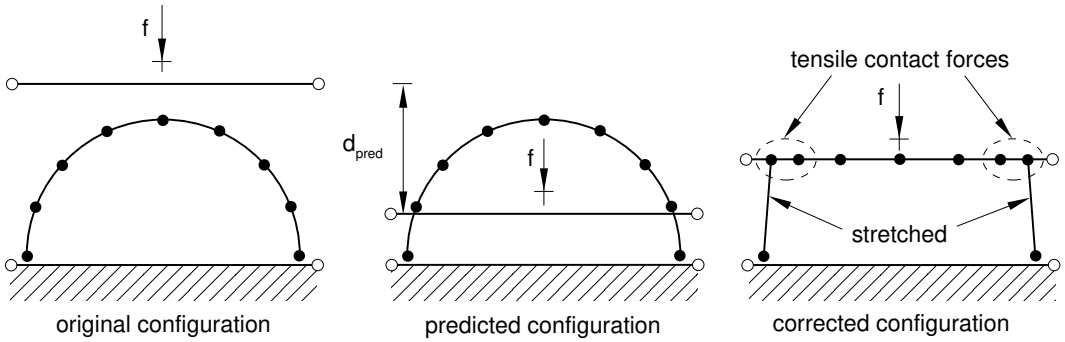


Figure 5: Undesirable numerical behavior of contact algorithm resulting from small rigid body mass.

With hard kinematic contact each secondary node that is penetrating its main surface in the predicted configuration will be brought to the position of its tracked point on the main surface in the corrected configuration, which, in this example, generates tensile contact forces at the outer secondary nodes of the contact region. This undesirable effect can be avoided by increasing the mass of the rigid body, which will reduce the predicted displacement increment. A small rigid body mass can also adversely affect penalty enforcement of contact because small penalty stiffnesses will be assigned.

Similar undesirable numerical behavior can occur for deformable-to-deformable contact if the nodal masses of the main nodes are orders of magnitude less than those of the secondary nodes. This problem can often be avoided in such cases by using the pure main-secondary algorithm with the main surface containing the more massive nodes.

Contact Noise Associated with Limited Computer Precision for Hard Contact

Some contact noise may occur with hard contact models because of limited computer precision. This noise is rarely significant in an analysis, but it may be noticeable at the beginning of an analysis if initial displacements are used to make the mesh comply with contact constraints. For example, if an adjustment of p_0 is made for an initial overclosure, a penetration of up to εp_0 may still exist in the first increment, where ε is the "machine epsilon" of the computer. The machine epsilon of a given computer is defined as the smallest positive number that can be added to 1 with the computed result being greater than 1; on most systems ε is approximately 6E−8 for single precision and 1E−16 for double precision. With the kinematic contact algorithm you can attribute initial accelerations of up to $\varepsilon p_0/\Delta t^2$ to limited machine precision, where Δt is the time increment. For a single precision analysis in which Δt=1E−6 sec, initial accelerations of up to 6E4 $\text{sec}^{-2} p_0$ can be attributed to limited machine precision. These accelerations are typically insignificant. They can be reduced by conducting the analysis with double precision or by specifying the nodal coordinates to be more compliant with contact constraints.

Finite-Sliding Contact near a Symmetry Plane

When a pure main-secondary contact constraint with finite sliding is defined near a symmetry plane in the main surface, the corner secondary node (node *A* in *Figure 6*) can, under some circumstances, slide freely along the symmetry plane without experiencing contact. If the main surface wraps around the corner (node 1), the secondary node *A* may "track" on the main segment (1–6) on the symmetry plane, rather than on main segment (1–2). The result may be an inaccurate representation of the contact constraint as shown by the shaded area.

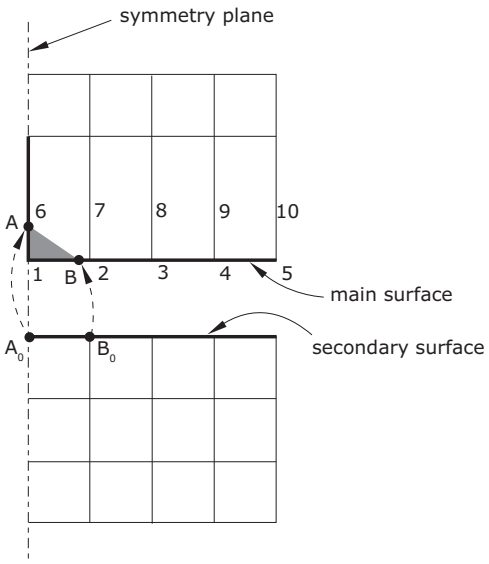


Figure 6: Contact near a symmetry plane. The main surface is wrapped around the corner.

If the main surface does not wrap around the corner (node 1 in *Figure 7*), the contact logic may give different results depending on how the symmetry boundary conditions have been defined for the main node 1 on the symmetry plane. If the symmetry boundary conditions on the main node are specified using boundary "type" format (i.e., XSYMM, YSYMM, or ZSYMM—see *Boundary Conditions*), the main surface is effectively extended beyond the symmetry plane (*Figure 7*); thus, the secondary node *A* will be detected as a "penetrated" node (penetrated by distance *a*). Therefore, a correcting force would be applied on secondary node *A* to push it below the main surface.

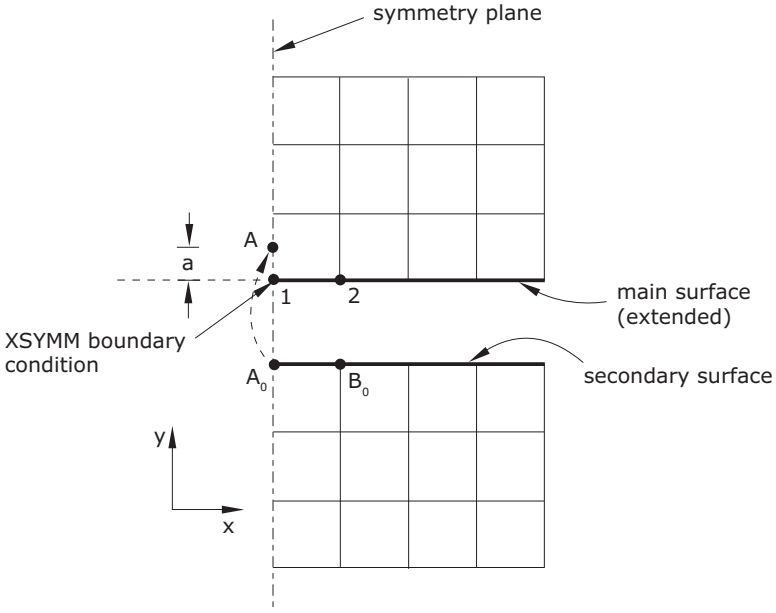


Figure 7: The main surface is extended across the symmetry plane because the symmetry boundary condition at node 1 is specified using boundary type XSYMM.

If the symmetry boundary conditions on the main node 1 are specified using "direct" format (i.e., specifying the components of translations and rotations that are fixed), the main surface is not extended beyond the symmetry plane (*Figure 8*) and it is possible that contact will not be enforced correctly.

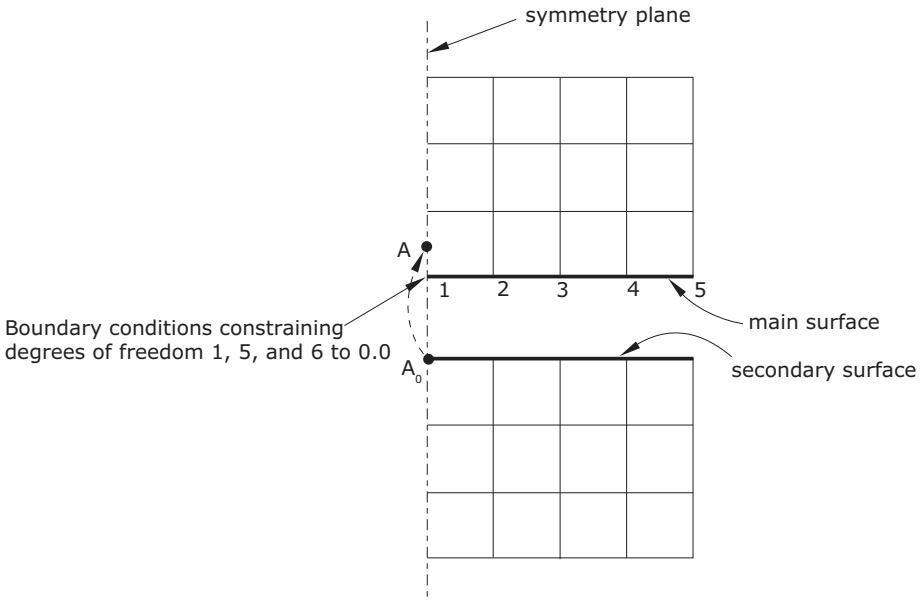


Figure 8: The main surface is not extended across the symmetry plane because the symmetry boundary conditions at node 1 are specified using direct format.

To ensure proper enforcement of finite-sliding contact near symmetry planes, use balanced main-secondary contact or use pure main-secondary contact without extending the surface onto the symmetry plane and use symmetry "type" boundary conditions on the perimeter of the main surface nodes as discussed above. Special consideration of small-sliding contact near a symmetry plane is discussed in *Contact Formulations for Contact Pairs in Abaqus/Explicit*.

Specifying Initial Clearance Values Precisely

You can define initial clearances and contact directions precisely for the nodes on the secondary surface (see *Specifying Initial Clearance Values Precisely*). The initial clearance or overclosure value calculated at every secondary node based on the coordinates of the secondary node and the main surface is overwritten by the value that you specify; the coordinates of the secondary nodes are not altered. This technique permits exact specification of initial clearances (and, possibly, contact directions) when they would not be computed accurately enough from the nodal coordinates; for example, if the initial clearance is very small compared to the coordinate values. It can be used only in small-sliding contact analyses (*Contact Formulations for Contact Pairs in Abaqus/Explicit*).

When the balanced main-secondary contact algorithm is invoked for the contact pair, the initial clearance values can be defined on one or both of the surfaces. Initial clearances defined on contact surfaces that act only as main surfaces will be ignored.

Visualizing the Precise Initial Clearances for Small-Sliding Contact Pairs

Abaqus/Explicit does not adjust the coordinates of the secondary surface when precise initial clearances are specified for small-sliding contact pairs (see *Contact Initialization for Contact Pairs in Abaqus/Explicit*). Therefore, the specified clearances cannot be seen in a postprocessor such as the Visualization module of Abaqus/CAE. Thus, depending on the initial geometry of the surfaces and the magnitude of the clearances or overclosures, the surfaces may appear open or closed in the postprocessor when they are actually just in contact.

Contact Elements in Abaqus/Standard

In this section:

- *Contact Modeling with Elements*
- *Gap Contact Elements*
- *Tube-to-Tube Contact Elements*
- *Slide Line Contact Elements*
- *Rigid Surface Contact Elements*

Contact Modeling with Elements

Overview

Abaqus/Standard offers a variety of contact elements that can be used when contact between two bodies cannot be simulated with the surface-based contact approach.
The surface-based contact approach is described in *Defining Contact Interactions*.

These elements include the following:

Gap contact elements

Mechanical and thermal contact between two nodes is modeled with gap elements (*Gap Contact Elements*). For example, these elements can be used to model the contact between a piping system and its supports. They can also be used to model an inextensible cable that supports only tensile loads.

Tube-to-tube contact elements

Contact between two pipes or tubes is modeled using tube-to-tube contact elements (*Tube-to-Tube Contact Elements*) in conjunction with slide lines. These elements can, for example, be used to simulate the process of running tubular components into an oil well (drill rod or J-tube analysis). They might also be used to simulate a catheter being inserted into a blood vessel.

Slide line contact elements

Finite-sliding contact between two axisymmetric structures that may undergo asymmetric deformations can be modeled using slide line contact elements (*Slide Line Contact Elements*) in conjunction with user-defined slide lines. Slide line elements can, for example, be used to model threaded connectors.

Rigid surface contact elements

Contact between an analytical rigid surface and an axisymmetric deformable body that may undergo asymmetric deformations can be modeled with rigid surface contact elements (*Rigid Surface Contact Elements*). For example, rigid surface contact elements might be used to model the contact between a rubber seal and a much stiffer structure.

References:

- *Defining Contact Interactions*

Gap Contact Elements

In this section:

- *Gap Contact Elements*
- *Gap Element Library*

Gap Contact Elements

Products: Abaqus/Standard

References:

- *Gap Element Library*
- *Defining Contact Interactions*
- **GAP*

Overview

Gap elements:

- allow for contact between two nodes;
- allow for the nodes to be in contact (gap closed) or separated (gap open) with respect to particular directions and separation conditions;
- are always defined in three dimensions but can also be used in two-dimensional and axisymmetric models;
- allow contact to be defined on any type of element, including substructures and user-defined elements;
- can be used to model contact in fixed or rotating directions;
- can be used to model node-to-node contact and thermal interactions in a fixed direction in space in coupled temperature-displacement simulations; and
- can be used to model node-to-node thermal interactions in heat transfer analyses.

A general discussion of contact modeling in Abaqus/Standard can be found in *Defining Contact Interactions*.

Choosing and Defining a Gap Element

GAPUNI elements model contact between two nodes when the contact direction is fixed in space. GAPCYL elements model contact between two nodes when the contact direction is orthogonal to an axis. GAPSPHER elements model contact between two nodes when the contact direction is arbitrary in space. GAPUNIT elements model contact and thermal interactions between two nodes when the contact direction is fixed in space. DGAP elements model thermal interactions between two nodes in heat transfer analysis.

Gap elements are defined by specifying the two nodes forming the gap and providing geometric data defining the initial state and, if necessary, the direction of the gap.

Defining the Gap Element's Properties

You must associate the gap behavior with a set of gap elements.

Input File Usage: **GAP*, ELSET=*element_set_name*

GAPUNI and GAPUNIT Elements

The contact behavior of the interface being modeled with GAPUNI and GAPUNIT elements is defined by the initial separation distance (clearance), d, of the gap and the contact direction, $\mathbf{n}$. In addition, GAPUNIT elements have temperature degrees of freedom that allow modeling of thermal interactions in coupled temperature-displacement analyses.

Clearance between GAPUNI Nodes

Abaqus/Standard defines the current clearance between two nodes of the gap, h, as

$$h = d + \mathbf{n} \cdot \left(\mathbf{u}^2 - \mathbf{u}^1\right),$$

where $\mathbf{u}^1$ and $\mathbf{u}^2$ are the total displacements at the first and the second node forming the GAPUNI element. *Figure 1* shows the configuration of the GAPUNI element. When h becomes negative, the gap contact element is closed and the constraint $h = 0$ is imposed.

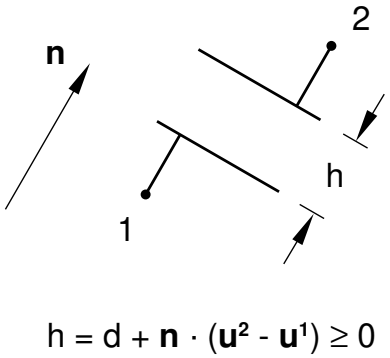


Figure 1: GAPUNI and GAPUNIT contact elements.

You specify a value for d. If you provide a positive value, the gap is open initially. If d=0, the gap is initially closed. If d is negative, the gap is considered overclosed at the start of the analysis and an initial interference fit problem is defined. Details about modeling interference fit problems with gap elements are discussed below.

Input File Usage: **GAP*
d

Specifying the Contact Direction

You can specify the contact direction. Otherwise, Abaqus/Standard will calculate the gap direction, $\mathbf{n}$, by using the initial positions of the two nodes forming the element, $\mathbf{X}^1$ and $\mathbf{X}^2$:

$$\mathbf{n} = \left(\mathbf{X}^2 - \mathbf{X}^1\right) / \left|\mathbf{X}^2 - \mathbf{X}^1\right|.$$

An error message is issued if $\mathbf{X}^2 = \mathbf{X}^1$ (if the two gap element nodes have the same initial coordinates). In this situation you must define $\mathbf{n}$. The normal $\mathbf{n}$ usually points from the first node of the element to the second, unless the gap is overclosed at the start of the analysis. In that case specify $\mathbf{n}$ so that the correct contact direction is used for the gap element.

If you specify the gap direction $\mathbf{n}$ rather than allowing Abaqus/Standard to calculate it, the contact calculations consider only $\mathbf{n}$, the displacements of the gap element's nodes, and the ordering of the nodes in the element definition: the initial coordinates of the nodes play no role in the calculations.

The orientation of $\mathbf{n}$ does not change during the analysis.

Input File Usage: **GAP*
, *X-direction cosine*, *Y-direction cosine*, *Z-direction cosine*

Local Basis System for GAPUNI Element Output

Abaqus/Standard reports the pressure transmitted across the gap and the shear stresses that are orthogonal to the contact direction as element output for GAPUNI elements. You must supply the contact area associated with these elements for Abaqus/Standard to compute the pressure and the shear stress values. It also reports the current clearance in the gap, h, and the relative motions of the GAPUNI nodes orthogonal to the contact direction. The relative motions and the shear stresses are reported in local surface directions that are formed using the standard Abaqus convention for defining directions on surfaces in space (see *Conventions*). The contact direction defines a surface in space on which the local axes are formed.

Input File Usage: **GAP*
, , , , cross-sectional area

GAPCYL Elements

GAPCYL elements can be used to model two very different contact situations: contact between two rigid tubes, where the smaller one is inside the larger tube, and contact between two rigid tubes along their external surfaces. Both cases are shown in *Figure 2*.

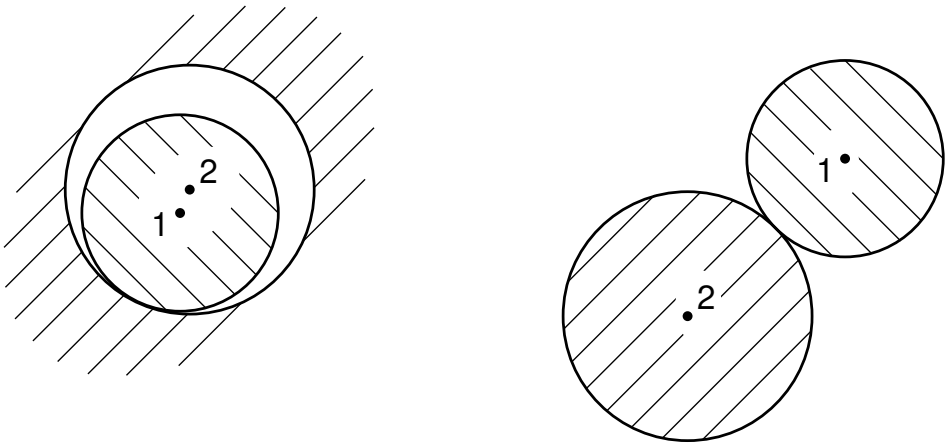


Figure 2: Gap clearance for GAPCYL/GAPSPHER contact elements.

The behavior of a GAPCYL element is defined by the initial separation distance between the nodes, d; the current positions of the element's node; and the axis of the GAPCYL element. The axis of the GAPCYL element defines the plane in which the contact direction, $\mathbf{n}$, lies. You specify d and the direction cosines of the GAPCYL element axis.

The value $\boldsymbol{d = 0}$ is not allowed: it would enforce the distance between the nodes to be exactly zero at all times, which does not correspond to a contact problem.

Input File Usage: **GAP*
d, X-direction cosine, Y-direction cosine, Z-direction cosine

Defining the Gap Clearance for Case 1 (When d Is Positive)

If d is positive, the GAPCYL element models contact between two rigid tubes of different diameter, where the smaller tube is located inside the larger tube (see Case 1 in *Figure 2*). In this case d is the maximum allowable separation. Each tube is represented by a node on its axis, with the axes connected by the GAPCYL element; and d corresponds to the difference between the radii of the tubes. The gap between the tubes closes when the two nodes become separated by more than d in any direction in the plane defined by the axis of the GAPCYL element.

Abaqus/Standard defines the current gap opening, h, in GAPCYL elements for Case 1 as

$$h = d - \left|\overline{\mathbf{x}}^2 - \overline{\mathbf{x}}^1\right| \qquad \text{with} \qquad \overline{\mathbf{x}}^N = \mathbf{x}^N - \mathbf{a}\left(\mathbf{a}\cdot\mathbf{x}^N\right),$$

where $\mathbf{x}^N$ is the current position of node N, d is the specified initial separation, and a is the axis of the GAPCYL element.

If the initial position of the tube axes is such that the distance between them is less than d, the GAPCYL element is open initially. If the distance is equal to d, the element is closed initially; and if the distance is greater than d, an initial overclosure (interference) is defined. Details about modeling interference fit problems with gap elements are discussed below.

Defining the Gap Clearance for Case 2 (When d Is Negative)

If d is negative, the GAPCYL element models external contact between two parallel rigid cylinders (see Case 2 in *Figure 2*). In this case $|d|$ is the minimum allowable separation of the nodes. Each cylinder is represented by a node on its axis connected by the GAPCYL element, and $|d|$ corresponds to the sum of the radii of the cylinders.

The gap closes when the two nodes approach each other to within $|d|$ in any direction in the plane defined by the axis of the GAPCYL element.

Abaqus/Standard defines the current gap opening, h, in GAPCYL elements for Case 2 as

$$h = \left|\overline{\mathbf{x}}^2 - \overline{\mathbf{x}}^1\right| - |d| \qquad \text{with} \qquad \overline{\mathbf{x}}^N = \mathbf{x}^N - \mathbf{a}\left(\mathbf{a}\cdot\mathbf{x}^N\right).$$

If the initial position of the cylinder axes is such that the distance between them is greater than $|d|$, the GAPCYL element is open initially. If the distance is equal to $|d|$, the element is closed initially; and if the distance is less than $|d|$, an initial overclosure (interference) is defined. Details about modeling interference fit problems with gap elements are discussed below.

Local Basis System for GAPCYL Element Output

Abaqus/Standard reports the pressure transmitted across the gap and the shear stresses that are orthogonal to the contact direction as element output for GAPCYL elements. You must supply the contact area associated with these elements for Abaqus/Standard to compute the pressure and the shear stress values. It also reports the current clearance in the gap, h, and the relative motions of the element's nodes that are orthogonal to the contact direction. The relative motions and the shear stresses are reported in local surface directions that are formed using the standard Abaqus convention for defining directions on surfaces in space (see *Conventions*). The contact direction defines a surface in space on which the local axes are formed, and the slip is calculated from the relative motions in the surface directions.

Abaqus/Standard updates the contact direction for GAPCYL elements based on the motion of the nodes forming the elements. However, the orientation of $\mathbf{a}$ is not updated during the analysis.

Input File Usage: **GAP*
, , , , *cross-sectional area*

GAPSPHER Elements

GAPSPHER elements can be used to model two very different contact situations: contact between two rigid spheres where the smaller sphere is inside the larger, hollow sphere, and contact between two rigid spheres along their external surfaces. Both cases are shown in *Figure 2*.

The behavior of a GAPSPHER element is defined by the minimum or maximum separation distance between the nodes, d, and the current positions of the element's nodes. You specify the minimum or maximum separation distance, d. The contact direction is defined by the current position of the nodes.

The value $d = 0$ is not allowed: it would enforce the distance between the nodes to be exactly zero at all times, which does not correspond to a contact problem.

Input File Usage: **GAP*
d

Defining the Gap Clearance for Case 1 (When d Is Positive)

If d is positive, the GAPSPHER element models contact between a rigid sphere inside another (larger) hollow rigid sphere (see Case 1 in *Figure 2*). In this case d is the maximum allowable separation of the nodes forming the gap. Each sphere is represented by a node at its center, with the centers connected by the GAPSPHER element; and d corresponds to the difference between the radii of the spheres. The gap closes when the two nodes become separated by more than d.

Abaqus/Standard defines the current gap opening, h, for Case 1 as

$$h = d - \left| \mathbf{x}^2 - \mathbf{x}^1 \right|,$$

with $\mathbf{x}^N$ the current position of node N and d the specified separation.

If the initial position of the tube axes is such that the distance between them is less than d, the GAPSPHER element is open initially. If the distance is equal to d, the element is closed initially; and if the distance is greater than d, an initial overclosure (interference) is defined. Details about modeling interference fit problems with gap elements are discussed below.

Defining the Gap Clearance for Case 2 (When d Is Negative)

If d is negative, the GAPSPHER element models external contact between two rigid spheres (see Case 2 in *Figure 2*). In this case $|d|$ is the minimum allowable separation of the nodes forming the gap. Each sphere is represented by a node at its center connected by the GAPSPHER element; and $|d|$ corresponds to the sum of the radii of the spheres. The gap closes when the two nodes approach each other to within $|d|$.

Abaqus/Standard defines the current gap opening, h, for Case 2 as

$$h = \left| \mathbf{x}^2 - \mathbf{x}^1 \right| - |d|.$$

If the initial position of the cylinder axes is such that the distance between them is greater than $|d|$, the GAPSPHER element is open initially. If the distance is equal to $|d|$, the element is closed initially; and if the distance is less than $|d|$, an initial overclosure (interference) is defined. Details about modeling interference fit problems with gap elements are discussed below.

Local Basis System for GAPSPHER Element Output

Abaqus/Standard reports the pressure transmitted across the gap and the shear stresses that are orthogonal to the contact direction as element output for GAPSPHER elements. You must supply the contact area associated with

these elements for Abaqus/Standard to compute the pressure and the shear stress values. It also reports the current clearance in the gap, h, and the relative motions of the element's node that are orthogonal to the contact direction. The relative motions and the shear stresses are reported in local surface directions that are formed using the standard Abaqus convention for defining directions on surfaces in space; see *Conventions*. The contact direction defines a surface in space on which the local axes are formed, and the slip is calculated from the relative motions in the surface directions.

Abaqus/Standard updates the contact direction for GAPSPHER elements based on the motion of the nodes forming the elements.

Input File Usage: **GAP*
, , , , *cross-sectional area*

DGAP Elements

DGAP elements are used to model thermal interactions between two nodes in heat transfer analyses. The behavior of the interaction being modeled is defined by the initial separation distance (clearance), d, of the gap.

Clearance between DGAP Nodes

Abaqus/Standard defines the clearance between two nodes of the gap, h, as

$$h = d.$$

Since there are no displacements in a heat transfer analysis, the clearance remains unchanged. The clearance is used only for clearance-dependent thermal interactions.

You specify a value for d. If you provide a positive value, the gap is open initially. If d=0, the gap is closed initially. If d is negative, the gap is considered overclosed but no interference fit is performed. The contact direction does not need to be specified: any contact direction specified is ignored in the analysis. You must supply the contact area associated with these elements for Abaqus/Standard to compute the heat flux value per unit area.

Input File Usage: **GAP*
d, , , , *cross-sectional area*

Defining Nondefault Mechanical Interactions with Gap Elements

The default mechanical interaction model for problems modeled with gap elements is “hard,” frictionless contact. You can assign optional mechanical interaction models. The following mechanical interaction models are available:

- Friction. See *Frictional Behavior* for details.
- Modified “hard” contact, softened contact, and viscous damping. See *Contact Pressure-Overclosure Relationships* and *Contact Damping* for details.

Defining Thermal Surface Interactions with GAPUNIT and DGAP Elements

You can assign thermal interaction models to these elements. The following thermal interaction models are available:

- Gap conduction.
- Gap radiation.
- Gap heat generation.

These thermal interaction models are discussed in *Thermal Contact Properties*.

Modeling Large Initial Interference with Gap Elements

Specifying a large negative initial overclosure (interference) may lead to convergence problems as Abaqus/Standard tries to resolve the overclosure in a single increment. You can prescribe an allowable interference to allow Abaqus/Standard to resolve the overclosure gradually. See *Modeling Contact Interference Fits in Abaqus/Standard* for more details on modeling interference fit problems.

Input File Usage: **CONTACT INTERFERENCE*, TYPE=ELEMENT

Gap Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the gap elements available in Abaqus/Standard.

References:

- *Gap Contact Elements*
- **GAP*

Element Types

Stress/Displacement Elements

GAPUNI	Unidirectional gap between two nodes
GAPCYL	Cylindrical gap between two nodes
GAPSPHER	Spherical gap between two nodes

Active Degrees of Freedom

1, 2, 3

Additional Solution Variables

Three additional variables relating to the contact and friction forces.

Coupled Temperature-Displacement Element

GAPUNIT	Unidirectional gap and thermal interactions between two nodes

Active Degrees of Freedom

1, 2, 3, 11

Additional Solution Variables

Three additional variables relating to the contact and friction forces.

Heat Transfer Element

DGAP Thermal interactions between two nodes

Active Degrees of Freedom

11

Additional Solution Variables

None.

Nodal Coordinates Required

For DGAP elements, and for GAPUNI and GAPUNIT if you specify the contact direction **n**, the nodal coordinates are not used in the contact calculations; however, it is useful to define the coordinates of the two nodes for plotting purposes.

GAPCYL and GAPSPHER: *X*, *Y*, *Z*

Element Property Definition

You can specify the initial clearance, the contact direction (normal to the interface), and the contact area.

For GAPUNI, GAPUNIT, and DGAP elements, a negative clearance indicates an initial overclosure.

For GAPCYL and GAPSPHER elements, specify the maximum separation as a positive number or the minimum separation as a negative number.

Input File Usage: **GAP*

Element-Based Loading

None.

Element Output

S11 Pressure transmitted between the surfaces. The pressure is defined as the force divided by the user-specified area.

S12 First frictional shear stress normal to the gap direction.

S13 Second frictional shear stress normal to the gap direction.

E11 Current opening h of the gap element.

E12 Relative displacement (“slip”) in the first direction orthogonal to the contact direction.

E13 Relative displacement (“slip”) in the second direction orthogonal to the contact direction.

Available for elements with temperature degrees of freedom.

HFL1 Heat flux across the interface in the contact direction.

The increments of shear slip are the relative displacement increments projected onto the two local directions that are orthogonal to the contact direction.

In two-dimensional or axisymmetric models when the contact direction is along the first axis (X or r), the active slip direction is E13 and the active shear stress is S13. In any other two-dimensional or axisymmetric case, the active slip direction is E12 and the active shear stress is S12.

Nodes Associated with the Element

Two nodes: the ends of the gap.

Tube-to-Tube Contact Elements

In this section:

- *Tube-to-Tube Contact Elements*
- *Tube-to-Tube Contact Element Library*

Tube-to-Tube Contact Elements

Products: Abaqus/Standard

References:

- *Defining Contact Interactions*
- *Tube-to-Tube Contact Element Library*
- **INTERFACE*
- **SLIDE LINE*

Overview

Tube-to-tube elements:

- model the finite-sliding interaction between two pipelines or tubes where one tube lies inside the other or between two tubes or rods that lie next to each other;
- are slide line contact elements, in the sense that they assume that the relative motion of the two tubes or pipes is predominantly along the line defined by the axis of one of the tubes (the relative rotations of the tube or pipe axis are assumed to be small);
- can be used with pipe, beam, or truss elements; and
- do not consider deformations of the tube or pipe cross-section.

Defining Contact Interactions contains a general discussion of contact modeling.

Typical Applications

The tube-to-tube contact elements can be used to model two specific classes of tube-to-tube contact problems: internal (tube within a tube) contact and external contact, where the two tubes are roughly parallel and contact each other along their outer surfaces.

Choosing an Appropriate Element

Use ITT21 elements with two-dimensional beam, pipe, or truss elements. Use ITT31 elements with three-dimensional beam, pipe, or truss elements. Each of these elements is defined by a single node.

Associating the Tube-to-Tube Contact Elements with a Slide Line

You must indicate which set of tube-to-tube contact elements will interact with a particular slide line. Details on defining slide lines are discussed below.

Input File Usage: **SLIDE LINE*, ELSET=*element_set_name*

Defining the Element's Section Properties

You must associate the geometric section properties with a set of tube-to-tube contact elements.

Input File Usage: **INTERFACE*, ELSET=*element_set_name*

Defining the Radial Clearance When Modeling Contact between a Pipe within Another Pipe

You define the radial clearance between the pipes. Give a positive value to model contact between two pipes when one pipe (the one with the tube-to-tube contact elements) lies inside of the other pipe. The value given is the difference between the inner radius of the outer pipe and the outer radius of the inner pipe.

Input File Usage: **INTERFACE*
radial clearance

Defining the Radial Clearance When Modeling Contact between the Outer Surfaces of Two Pipes

You can model external tube-to-tube contact by specifying a negative value for the radial clearance. The magnitude of the value must be the sum of the outer radii of the two pipes or rods.

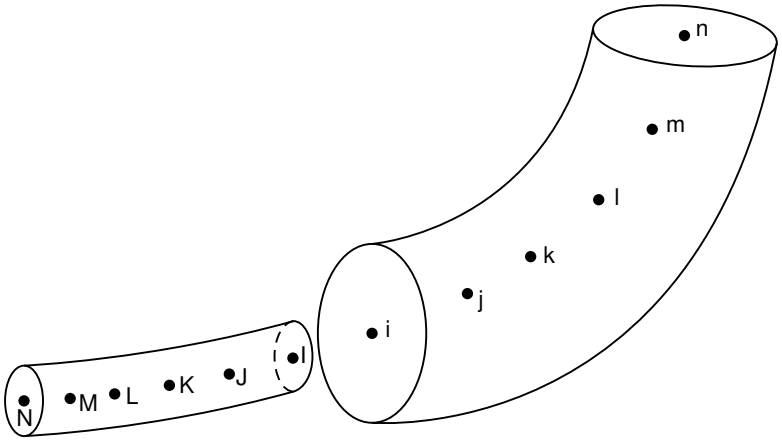
Local Basis for Contact Output Variables

The element output variables for ITT elements are given in a local basis system associated with the slide line.

The first tangent vector, $\mathbf{t_1}$, is defined by the sequence of the nodes forming the slide line. The direction of contact, $\mathbf{n}$, is the normal to the slide line that points toward the nodes of the ITT elements. For ITT31 elements Abaqus/Standard forms a second tangent vector, $\mathbf{t_2}$, that is orthogonal to both $\mathbf{t_1}$ and $\mathbf{n}$. As the elements move, the local basis system will rotate with the axis of the slide line.

Choosing Which Pipe (Beam or Truss) Will Have the Slide Line

In the case of internal tube-to-tube contact, the slide line can be placed on the inner tube or the outer tube. Generally the slide line should be associated with the outer tube (see *Figure 1*); however, if the inner tube is stiffer than the outer tube, the slide line should be attached to the inner tube.



Nodes i, j, k, l, m, and n are specified in that order, thereby identifying a slide line progressing from i to node n. These nodes must lie on the outer tube. ITT-type elements are defined on nodes I, J, K, ... and interact with the slide line.

Figure 1: Internal tube-to-tube contact example.

If contact occurs between the exterior surface of the tubes, the slide line should be associated with the stiffer tube if the materials or tube radii are different or with the tube with the coarser mesh if they are the same.

Defining the Slide Line

You can specify the nodes that make up the slide line, or they can be generated as described below. If you choose to specify the nodes directly, you must specify them in a sequence that defines a continuous slide line. The nodal sequence defines a tangent vector $\mathbf{t}_1$ for the slide line. The slide line must be made up of linear segments.

Input File Usage: **SLIDE LINE*, ELSET=*element_set_name*, TYPE=LINEAR
first node number, second node number, etc.

Generating the Slide Line Nodes

Alternatively, you can indicate that the slide line nodes should be generated and specify only a first node number, a last node number, and an increment between node numbers.

Input File Usage: **SLIDE LINE*, GENERATE
first node number, last node number, increment between node numbers

Smoothing the Slide Line

Convergence is often improved by smoothing the discontinuities in surface tangents between slide line segments, thereby providing a smoothly varying tangent along the slide line. For details about smoothing slide lines, see *Contact Formulations in Abaqus/Standard*.

Defining Nondefault Mechanical Surface Interactions with Tube-to-Tube Contact Elements

By default, Abaqus/Standard uses "hard," frictionless contact with tube-to-tube contact elements. You can assign optional mechanical surface interaction models. The following mechanical surface interaction models are available:

- Friction. See *Frictional Behavior* for details.
- Modified "hard" contact, softened contact, and viscous damping. See *Contact Pressure-Overclosure Relationships* and *Contact Damping* for details.

Tube-to-Tube Contact Element Library

Products: Abaqus/Standard

Overview

References:

- *Tube-to-Tube Contact Elements*
- **INTERFACE*
- **SLIDE LINE*

Element Types

ITT21	Tube-to-tube element for use with two-dimensional beam and pipe elements
ITT31	Tube-to-tube element for use with three-dimensional beam and pipe elements

Active Degrees of Freedom

ITT21: 1, 2

ITT31: 1, 2, 3

Additional Solution Variables

ITT21: Two additional variables relating to the contact forces.

ITT31: Three additional variables relating to the contact forces.

Nodal Coordinates Required

ITT21: *X*, *Y*

ITT31: *X*, *Y*, *Z*

Element Property Definition

Input File Usage: Use the following option to identify the second (outer) pipe with which the specified ITT contact elements on the first (inner) pipe can interact:

**SLIDE LINE*

Use the following option to give the radial clearance between the pipes as a positive number when modeling a tube sliding within another tube:

**INTERFACE*

When the elements are modeling contact between the exterior surfaces of two pipes, the sum of the external radii of the pipes is given as a negative number.

Element-Based Loading

None.

Element Output

Stress Components

S11 Normal component of the force between the two pipes.

S12 Shear force between the two pipes, parallel to the axis of the second (outer) pipe.

S13 Shear force between the two pipes, normal to the contact direction and to the axis of the second (outer) pipe (for ITT31 only).

Strain Components

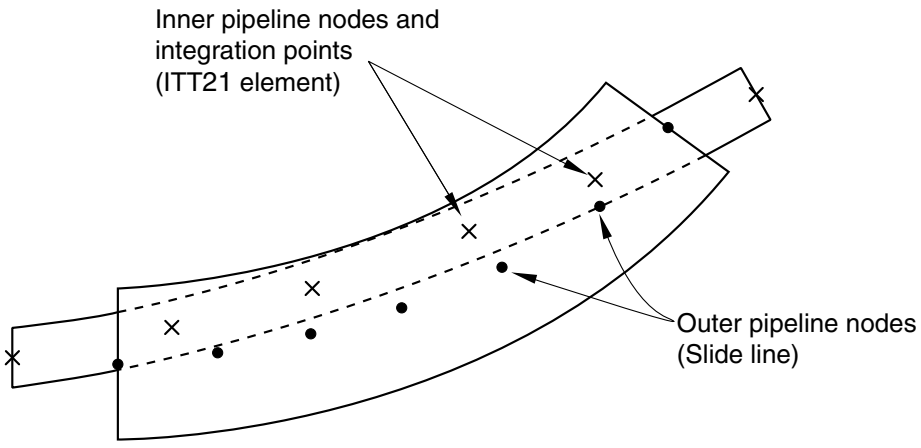
E11 Overclosure of the surfaces in the direction normal to the tangent to the centerline of the second (outer) pipe.

E12 Accumulated relative tangential motion between the two pipes, parallel to the axis of the second (outer) pipe.

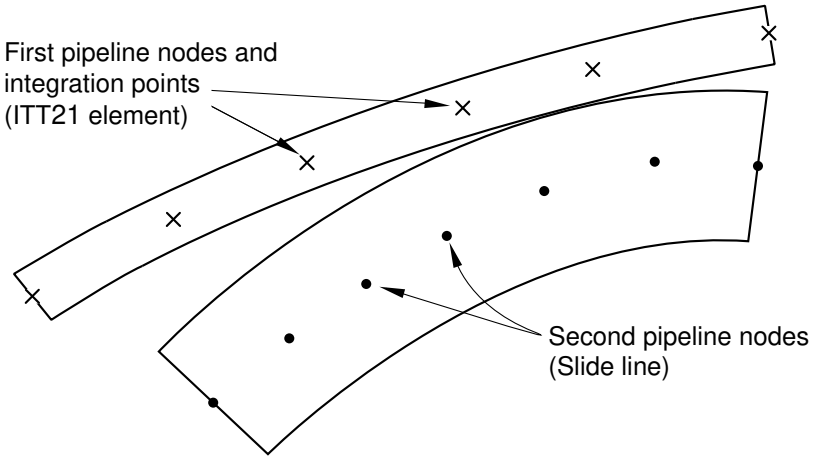
E13 Accumulated relative tangential motion between the two pipes, normal to the contact direction and to the axis of the second (outer) pipe (for ITT31 only).

Node Ordering and Integration Point Numbering

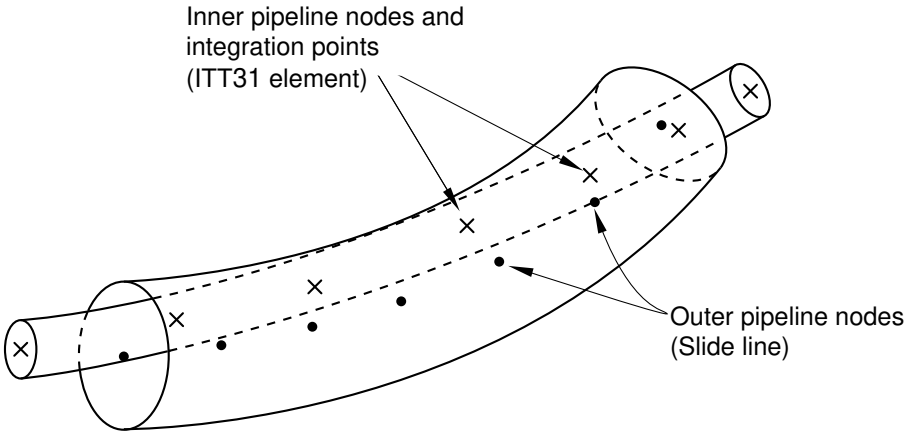
2D Internal Tube Contact



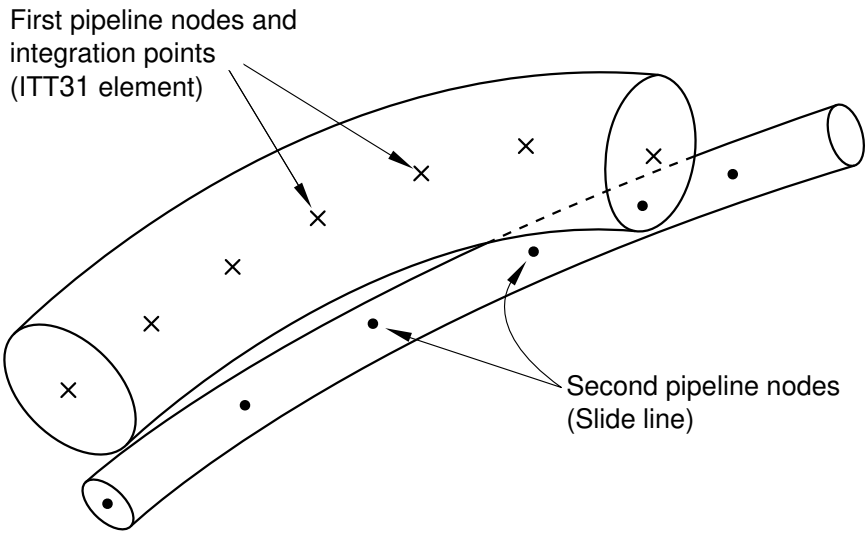
2D External Tube Contact



3D Internal Tube Contact



3D External Tube Contact


First pipeline nodes and
integration points
(ITT31 element)
Second pipeline nodes
(Slide line)

Slide Line Contact Elements

In this section:

- *Slide Line Contact Elements*
- *Axisymmetric Slide Line Element Library*

Slide Line Contact Elements

Products: Abaqus/Standard

References:

- *Axisymmetric Slide Line Element Library*
- *Defining Contact Interactions*
- **INTERFACE*
- **SLIDE LINE*

Overview

Slide line elements:

- can model the finite-sliding interaction between two deforming bodies when the sliding occurs along a line (“slide line”) that lies in a specific plane;
- assume that tangential motions orthogonal to a slide line are zero or small (Abaqus/Standard treats such motions as being infinitesimal);
- can be used with axisymmetric stress/displacement elements;
- are recommended for specific applications, such as when a contact surface is the surface of a substructure or when CAXA or SAXA elements are involved in contact;
- are available for first- and second-order elements; and
- use the same “main-secondary” concepts for enforcing contact constraints seen in surface-based contact.

For a general discussion of contact modeling, see *Defining Contact Interactions*.

Modeling Contact between Deformable Bodies with Slide Lines

Determining the location of the areas of contact and the surface tractions between contacting structures are common goals of Abaqus simulations (see *Figure 1*). Slide lines and slide line contact elements can provide this information for simulations where both structures are deformable and the finite sliding of the structures occurs along well-defined lines.

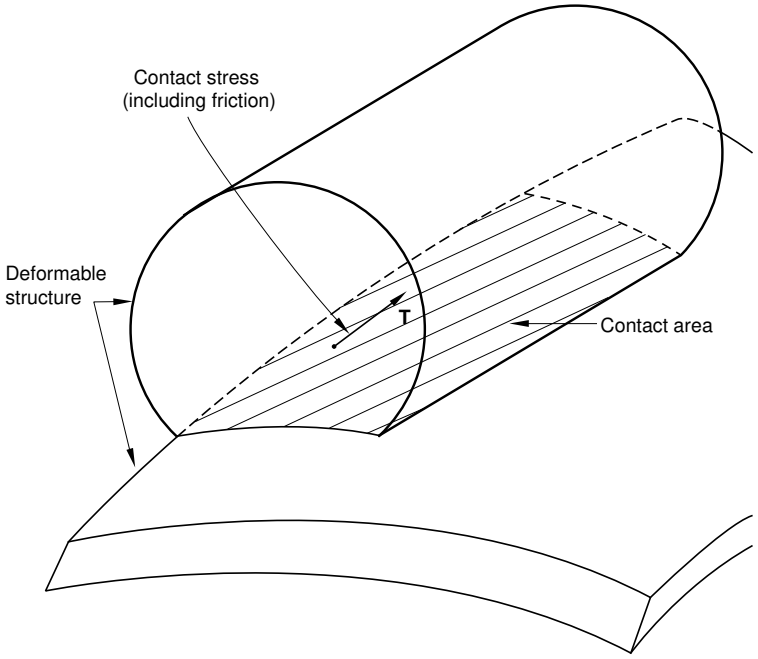


Figure 1: Interaction between deformable structures.

Local Basis System for Contact Stresses and Relative Motions of the Bodies

Abaqus/Standard reports the contact stresses between the bodies and the relative motions of the bodies in a local basis system that is attached to the slide line surface. The local basis system is defined by the normal to the slide line, $\mathbf{n}$, and two orthogonal local tangent directions, $\mathbf{t_1}$ and $\mathbf{t_2}$ (see *Figure 2*).

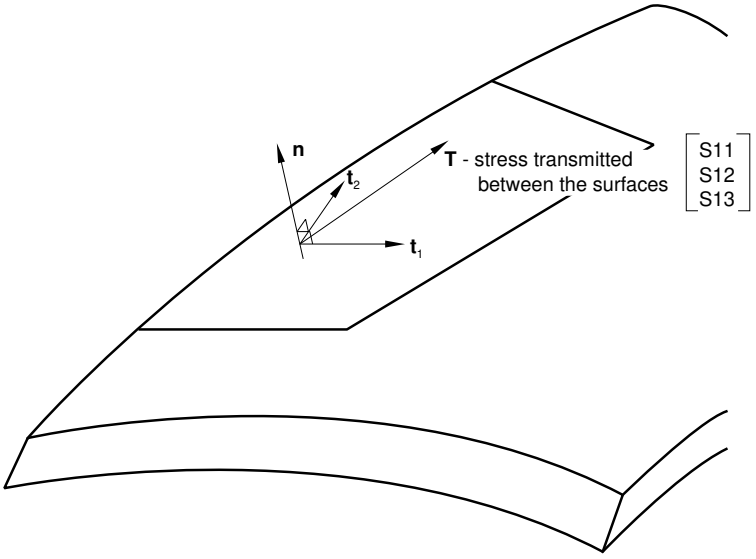


Figure 2: Local system for interface contact normal and shear traction.

Defining the Local Basis System

The sequence of the nodes forming the slide line defines the tangent, $\mathbf{t}$. The plane formed by the slide line normal, $\mathbf{n}$, and $\mathbf{t}$ is called the contact plane. Abaqus/Standard defines the slide line normal as $\mathbf{n} = \mathbf{s} \times \mathbf{t}$ (see *Figure 3*), where $\mathbf{s} = (0, 0, 1)$ is the vector that is orthogonal to the contact plane.

As shown in *Figure 3*, a slide line is created using nodes i, j, k, …, p, which are specified in that order, thereby identifying the slide line tangent. Nodes I, J, K, …, N are the nodes of the slide line elements that are associated with this slide line. The slide line normal $\mathbf{n}$ is defined by specifying $\mathbf{s}$, the normal to the contact plane.

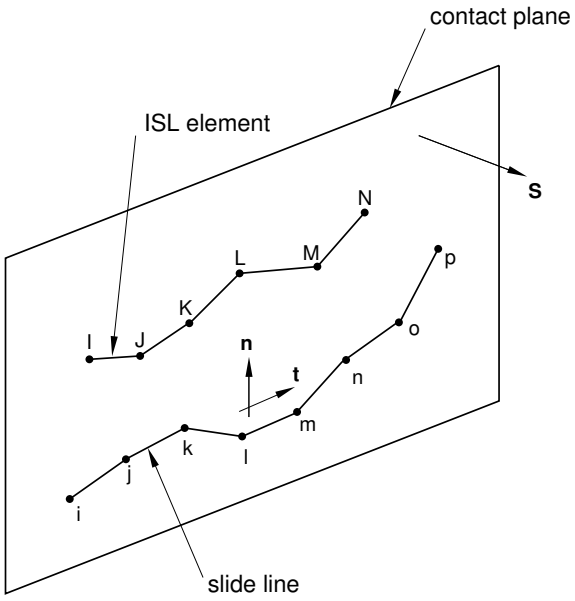


Figure 3: Defining the local basis for a slide line.

The tangent to the slide line coincides with the first local tangent direction, $\mathbf{t}_1$, of the local basis system. The second local tangent direction, $\mathbf{t}_2$, is in the opposite direction of $\mathbf{s}$.

The Main-Secondary Concept for Slide Lines and Slide Line Elements

When creating a model that contains slide line elements, it is useful to remember that Abaqus/Standard uses a strict "main-secondary" concept to enforce the contact constraints. The slide line contact elements form the "secondary" surface. The nodes that you specify to define the slide line define the "main" surface. The nodes of the slide line contact elements are constrained not to penetrate the main surface.

The considerations for choosing the main and secondary surfaces are the same regardless of whether surfaces or elements are used to define contact. The main surface should be chosen as the surface of the stiffer body if the materials are different or as the surface with the coarser mesh. If the materials and mesh density are the same on both surfaces, the choice is arbitrary.

Defining the Slide Line (Main Surface)

You can specify the nodes that make up the slide line, or they can be generated as described below. If you choose to specify the nodes directly, you must specify them in a sequence that defines a continuous slide line. The nodal sequence defines a tangent vector, $\mathbf{t}$, for the slide line. The slide line can be made up of linear or parabolic segments, depending on whether the model is made up of first-order or second-order elements. In either case convergence may be improved by smoothing the slide line.

Defining a Linear Slide Line

When the surfaces of the bodies are meshed with first-order elements, define a slide line made up of linear element segments. As shown in *Figure 4*, nodes i, j, k, …, p are specified in that order, thereby identifying a slide line progressing from i through p. Nodes I, J, K, …, N are the nodes of the ISL-type elements that are associated with this slide line.

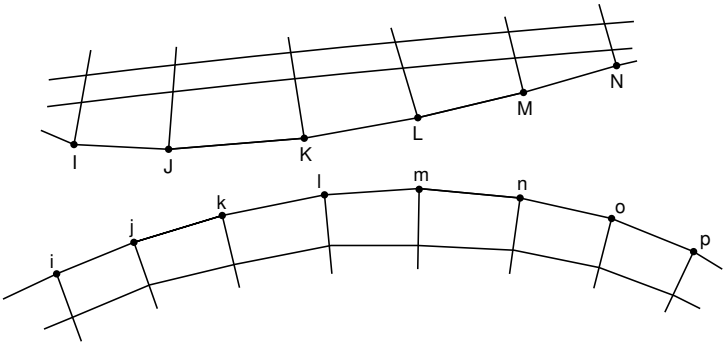


Figure 4: First-order (linear) slide line example.

Input File Usage: **SLIDE LINE*, ELSET=*element_set_name*, TYPE=LINEAR
first node number, second node number, etc.

Defining a Parabolic Slide Line

When the surfaces of the bodies are meshed with second-order elements, define a slide line made up of second-order element segments. In this case the slide line should consist of an odd number of nodes. As shown in *Figure 5*, nodes *i*, *j*, *k*, …, *u* are specified in that order, thereby identifying a slide line progressing from *i* through *u*. Nodes *I*, *J*, *K*, …, *O* are the nodes of the ISL-type elements that are associated with this slide line.

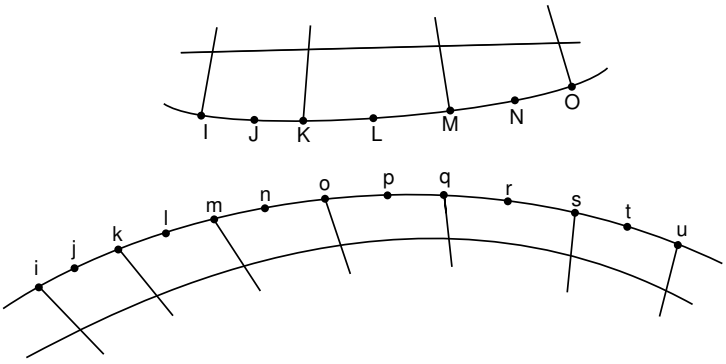


Figure 5: Second-order (parabolic) slide line example.

Input File Usage: **SLIDE LINE*, ELSET=*element_set_name*, TYPE=PARABOLIC
first node number, second node number, etc.

Generating the Slide Line Nodes

Alternatively, you can indicate that the slide line nodes should be generated and specify only a first node number, a last node number, and an increment between node numbers.

Input File Usage: **SLIDE LINE*, ELSET=*element_set_name*, GENERATE
first node number, last node number, increment between node numbers

Smoothing the Slide Line

Convergence is often improved by smoothing the discontinuities in surface tangents between slide line segments, thereby providing a smoothly varying tangent along the slide line. For details about smoothing slide lines, see *Contact Formulations in Abaqus/Standard*.

Defining Slide Line Elements (Secondary Surface)

Many finite-sliding contact simulations can use the surface-based contact approach, described in *Defining Contact Interactions*, to define the model. Axisymmetric stress/displacement and coupled temperature-displacement slide line elements are recommended only for specific applications, such as when a contact surface is the surface of a substructure or when CAXA or SAXA elements are involved in contact (see *Contact Modeling If Asymmetric-Axisymmetric Elements Are Present*).

The slide line contact elements define the secondary surface. The contact area associated with each node on the secondary surface is calculated using the current length of the slide line contact element and the constant "width" assigned to the element, which depends on the underlying finite elements.

Associating the Slide Line Elements with a Slide Line

You must associate the slide line with a set of slide line contact elements. Details on defining slide lines are discussed below.

Input File Usage: **SLIDE LINE*, ELSET=*element_set_name*

Defining the Slide Line Element's Section Properties

You must associate the section properties with a set of slide line elements.

There are no section data for axisymmetric slide line elements.

Input File Usage: **INTERFACE*, ELSET=*element_set_name*

Defining Nondefault Mechanical Surface Interactions with Slide Line Elements

By default, Abaqus/Standard uses "hard," frictionless contact with slide line elements. You can assign optional mechanical surface interaction models. The following mechanical surface interaction models are available:

- Friction. See *Frictional Behavior* for details.
- Modified "hard" contact, softened contact, and viscous damping. See *Contact Pressure-Overclosure Relationships* and *Contact Damping* for details.

Obtaining the "Maximum Torque" That Can Be Transmitted across Axisymmetric Slide Lines

When modeling contact with slide lines with axisymmetric elements (type CAX and CGAX elements), Abaqus/Standard can calculate the maximum torque that can be transmitted across the axisymmetric slide lines. This capability is often of interest when modeling threaded connectors. The maximum torque, T, is defined as

$$T = \int\int r^2 p \, ds \, d\theta,$$

where p is the pressure transmitted across the interface, r is the radius to a point on the interface, and s is the current distance along the interface in the r–z plane. This definition of "torque" effectively assumes a friction coefficient of unity.

You can request that this torque output be written to the data (`.dat`) file. The data are provided for every slide line in the model. You can specify the output frequency to limit how often Abaqus/Standard writes this output to the data file. The default output frequency is 1.

For surface-based contact with axisymmetric elements, output variable CTRQ provides functionality similar to this torque output request (see *About Contact Pairs in Abaqus/Standard*).

Input File Usage: **TORQUE PRINT*, FREQUENCY=*n*

Axisymmetric Slide Line Element Library

Products: Abaqus/Standard

Overview

References:

- *Slide Line Contact Elements*
- **INTERFACE*
- **SLIDE LINE*

Element Types

ISL21A	2-node element for use with first-order axisymmetric elements
ISL22A	3-node element for use with second-order axisymmetric elements

Active Degrees of Freedom

1, 2 at the nodes

Additional Solution Variables

Two additional variables at each node relating to the contact stresses.

Nodal Coordinates Required

r, z

Element Property Definition

Input File Usage: Use the following option to identify the slide line (main surface) with which the slide line elements interact:

**SLIDE LINE*

Use the following option to define the slide line element's section properties:

**INTERFACE*

Element-Based Loading

None.

Element Output

Stress Components

S11 Pressure between the node on the body and the slide line with which it interacts.

S12 Shear stress between the node on the body and the slide line with which it interacts.

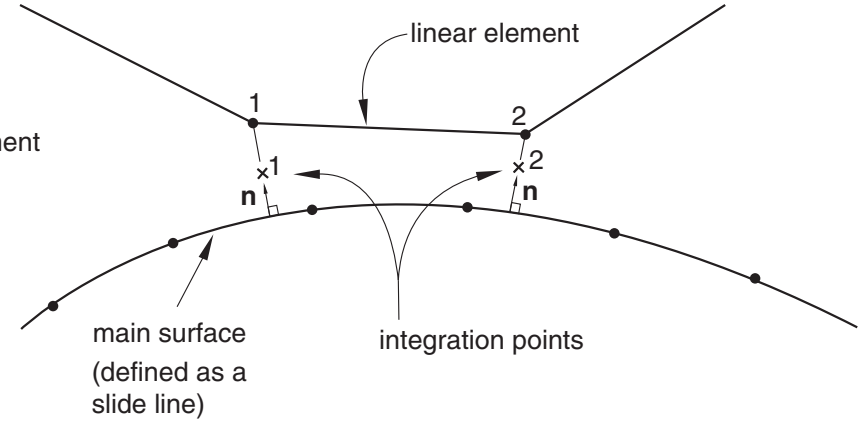
Strain Components

E11 Separation between the node on the body and the slide line.

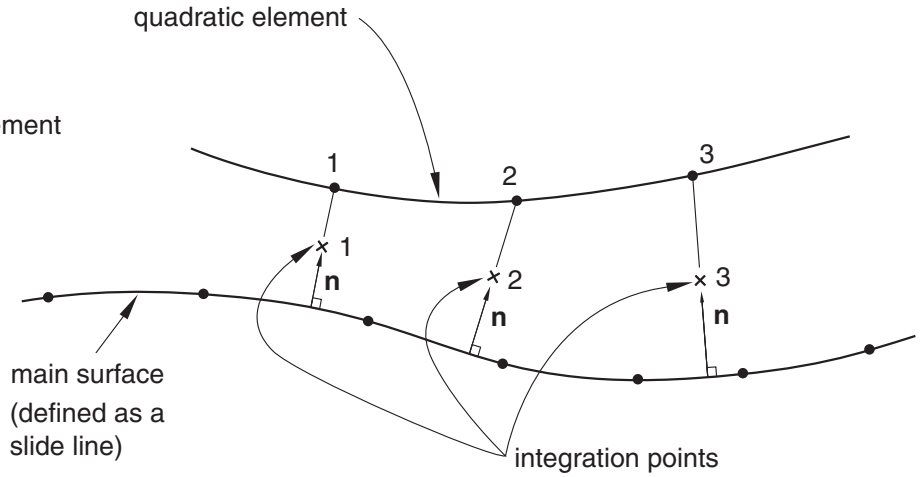
E12 Accumulated relative tangential displacement between the node on the body and the slide line.

Node Ordering and Integration Point Numbering

2 - node element



3 - node element



Rigid Surface Contact Elements

In this section:

- *Rigid Surface Contact Elements*
- *Axisymmetric Rigid Surface Contact Element Library*

Rigid Surface Contact Elements

Products: Abaqus/Standard

References:

- *Defining Contact Interactions*
- *Axisymmetric Rigid Surface Contact Element Library*
- *Analytical Rigid Surface Definition*
- **INTERFACE*
- **RIGID SURFACE*

Overview

Rigid surface contact elements:

- can be used to model contact between a rigid surface and a deformable body;
- are needed only for several special-purpose applications, such as when a substructure contacts a rigid surface or when CAXA or SAXA element types are involved in contact;
- can be used in both geometrically linear and nonlinear simulations; and
- use the same "main-secondary" concepts for enforcing contact constraints that are used in the surface-based contact capability in Abaqus/Standard.

For most problems the surface-based contact capability described in *Defining Contact Interactions* provides a more direct and general method for modeling contact between a rigid surface and a deformable body.

Modeling Contact between Rigid Surfaces and Rigid Surface Contact Elements

Determining the location of the areas of contact and the surface tractions between contacting structures are common goals of Abaqus simulations. Rigid surface contact elements can be used to model contact when one of the structures is assumed to be rigid. These elements need to be used only for specific applications, outlined below, because the surface-based contact definitions in Abaqus can be used for most simulations.

Modeling Contact with Axisymmetric Rigid Surface Contact Elements

Axisymmetric rigid surface contact elements should be used only in the following specific applications:

- when the deformable surface is on a substructure (see *Contact Modeling If Substructures Are Present*), or
- when CAXA or SAXA elements are involved in contact (see *Contact Modeling If Asymmetric-Axisymmetric Elements Are Present*).

Other planar, axisymmetric, or three-dimensional problems should use the surface-based contact capability.

Local Basis System for Contact Stress and Relative Motions of the Surfaces

Abaqus/Standard reports the contact stresses between the bodies and the relative motions of the bodies in a local basis system that is attached to the rigid surface. The normal $\mathbf{n}$ to the rigid surface, which is also the contact direction, is defined when the rigid surface is created. For details, see *Analytical Rigid Surface Definition*. In axisymmetric problems Abaqus/Standard defines the first local tangent to lie in the plane of the model and the second orthogonal to this plane.

The Main-Secondary Concept for Rigid Surface Contact Elements

Rigid surface contact elements use a “main-secondary” concept to enforce the contact constraints. The rigid surface contact elements form the “secondary” surface, and the nodes of these elements are constrained not to penetrate into the rigid (“main”) surface.

Defining the Rigid Surface

You define the analytical rigid surface using the methods described in *Defining Analytical Rigid Surfaces When Drag Chain or Rigid Surface Elements Are Used*.

Assigning a Rigid Body Reference Node to the Rigid Surface

The motion of a rigid surface is controlled by the motion of a single node, referred to as the rigid body reference node, that is associated with the rigid surface. When rigid surface contact elements are used in a model, the rigid body reference node is identified when defining the IRS elements (see below for details).

Defining the Rigid Surface Contact Elements

The rigid surface contact elements define the secondary surface. They also define the rigid body reference node for the rigid surface with which they interact. All IRS elements identify the rigid body reference node by including its node number as the last node in their connectivity. The nodes on the deformable body that form the IRS elements are always given first.

In a model defined in terms of an assembly of part instances, the rigid surface definition and the reference node must appear inside the same part definition as the rigid surface contact elements.

Example

For example, the following input would be used to define IRS elements 1 and 2 that consist of two nodes on the deformable body and assign node 1000 as the rigid body reference node:

```
*ELEMENT, TYPE=[IRS21A], ELSET=element_set_name
1, 10, 11, 1000
2, 11, 12, 1000
*RIGID SURFACE, ELSET=element_set_name
```

A similar input structure is used for IRS22A elements.

Associating an Analytical Rigid Surface with a Set of Rigid Surface Contact Elements

You must identify the set of rigid surface contact elements that interact with a particular rigid surface.

Input File Usage: **RIGID SURFACE*, ELSET=*element_set_name*

Defining the Rigid Surface Element's Section Properties

You must associate the section properties with a set of rigid surface contact elements.

There are no section data for axisymmetric rigid surface contact elements.

Input File Usage: **INTERFACE*, ELSET=*element_set_name*

Defining Nondefault Mechanical Surface Interactions with Rigid Surface Contact Elements

By default, Abaqus/Standard uses a “hard,” frictionless mechanical surface interaction model with rigid surface contact elements. You can assign optional mechanical surface interaction models. The following mechanical surface interaction models are available:

- Friction. See *Frictional Behavior* for details.
- Modified “hard” contact, softened contact, and viscous damping. See *Contact Pressure-Overclosure Relationships* and *Contact Damping* for details.

Axisymmetric Rigid Surface Contact Element Library

Products: Abaqus/Standard

Overview

This section provides a reference to the axisymmetric rigid surface contact elements available in Abaqus/Standard.

References:

- *Analytical Rigid Surface Definition*
- *Rigid Surface Contact Elements*
- **RIGID SURFACE*
- **INTERFACE*

Element Types

IRS21A	Axisymmetric rigid surface contact element for use with first-order axisymmetric elements
IRS22A	Axisymmetric rigid surface contact element for use with second-order axisymmetric elements

Active Degrees of Freedom

1, 2 at each node except the last node

1, 2, 6, the motion of the rigid body reference node, at the last node

Additional Solution Variables

Two additional variables at each node relating to the contact stresses.

Nodal Coordinates Required

r, z

Element Property Definition

Input File Usage: Use the following option to define the surface with which the elements interact:

**RIGID SURFACE*

Use the following option to define the rigid surface element's section properties:

**INTERFACE*

Element-Based Loading

None.

Element Output

S11 Pressure between the element and the rigid surface in the direction of the normal to the rigid surface.

S12 Shear component of the stress between the element and the rigid surface in the direction of the tangent to the rigid surface.

E11 Separation of the surfaces in the direction of the normal to the rigid surface at the closest point of the surface to the integration point on the element.

E12 Accumulated relative tangential displacement of the surfaces.

Node Ordering on Elements

The first two nodes in IRS21A and the first three nodes in IRS22A are on the deforming mesh. The last node is the rigid body reference node that defines the motion of the rigid body.

Numbering of Integration Points for Output

The integration points are located at the nodes that lie on the surface of the deforming model and are numbered correspondingly.

Cavity Radiation in Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References:

- *Defining an Analysis*
- *About Heat Transfer Analysis Procedures*
- **CAVITY DEFINITION*
- **COUPLED THERMAL-ELECTRICAL*
- **CYCLIC*
- **EMISSIVITY*
- **HEAT TRANSFER*
- **MOTION*
- **PERIODIC*
- **PHYSICAL CONSTANTS*
- **RADIATION FILE*
- **RADIATION PRINT*
- **RADIATION OUTPUT*
- **RADIATION SYMMETRY*
- **RADIATION VIEW FACTOR*
- **REFLECTION*
- **SURFACE*
- **SURFACE PROPERTY*
- **VIEW FACTOR OUTPUT*
- *Cavity radiation*
- *Defining a cavity radiation interaction*
- *Defining a cavity radiation interaction property*

Overview

Abaqus/Standard provides a cavity radiation capability for modeling heat transfer effects due to radiation in enclosures. This cavity radiation functionality:

- can be included in heat transfer analysis problems without deformation (*Uncoupled Heat Transfer Analysis* and *Coupled Thermal-Electrical Analysis*);
- is provided for two-dimensional, three-dimensional, and axisymmetric cases;
- accounts for symmetries, surface blocking, and surface motion within cavities; and
- can include closed cavities or open cavities (implying that some radiation takes place to an exterior medium).

Cavity radiation equations are not symmetric; therefore, the nonsymmetric matrix storage and solution scheme is invoked automatically in models that include cavity radiation (see *Cavity radiation* and *Defining an Analysis*). Each cavity defines a view factor matrix involving the geometric relations between the surfaces in the enclosure. These

matrices may be updated a number of times during the analysis (due to moving surfaces in the cavity). Therefore, large cavity radiation problems may be computationally expensive. Instead, you should consider using:

- gap radiation (see *Thermal Contact Properties*) for modeling radiation between closely spaced surfaces;
- average-temperature radiation conditions for modeling enclosures that are approximately isothermal, with constant emissivity, and do not require blocking or reflection considerations (see *Thermal Loads*); or
- parallel cavity decomposition for parallel calculation of view factors and solution of the radiative heat transfer equations (see *Decomposing Large Cavities in Parallel* below).

Defining a Cavity Radiation Problem

Since cavity radiation effects are calculated only in heat transfer and coupled thermal-electrical procedures, the only kind of thermal-stress analysis that can include these effects is sequentially coupled thermal-stress analysis (see *Sequentially Coupled Thermal-Stress Analysis*). Moreover, unless you allow cavity parallel decomposition (see *Decomposing Large Cavities in Parallel* below), there is a software limit of 16,000 nodes and facets in Abaqus/Standard. In addition, there is a limit that you can define at most 100 cavities in a single analysis.

Model Definition

When you define the model for a cavity radiation problem, you must:

1. define all of the surfaces in the cavity (see *Defining Surfaces*);
2. define the radiation properties of each surface (i.e., the emissivity) and the physical constants (see *Defining Surface Radiation Properties*); and
3. construct cavities from the surfaces (see *Constructing a Cavity*).

History Definition

In the first step of a cavity radiation analysis you must associate with each cavity a radiation view factor definition, which controls the calculation of view factors for the cavity. You then may:

1. define cavity symmetries, if any (see *Defining Cavity Symmetries*);
2. prescribe the motion of surfaces (see *Prescribing Motion during a Cavity Radiation Analysis*);
3. define boundary conditions such as temperature and forced convection (see *Boundary Conditions*);
4. control the cavity radiation and view factor calculations in each step (the specifications from the previous step are used if they are not redefined in a step; see *Controlling View Factor Calculation during the Analysis*);
5. request output of heat transfer variables to the data and results files (see *Requesting Surface Variable Output*); and
6. request output of the radiation view factor matrices (see *Writing the View Factor Matrices to the Results File*).

If any of the above are included in your analysis, they must be defined within a heat transfer or coupled thermal-electrical step definition.

Defining Surfaces

Cavities are defined in Abaqus/Standard as collections of surfaces, which are composed of facets. In axisymmetric and two-dimensional cases a facet is a side of an element; in three-dimensional cases a facet is a face of a solid element or a surface of a shell element. Rigid surfaces cannot be used in cavity radiation problems.

Surfaces are defined as described in *Element-Based Surface Definition*. You may associate each surface with a surface property definition as part of the surface option, or you may associate surfaces with surface properties as part of the cavity definition option. The surface properties are defined as described below.

Input File Usage: Use the following option to define a surface with a surface property for use in a cavity radiation analysis:

**SURFACE*, TYPE=ELEMENT, NAME=*surface_name*, PROPERTY=*property_name*

Use the following option to define a surface for use in a cavity radiation analysis in which surface properties are defined as part of the cavity definition:

**SURFACE*, TYPE=ELEMENT, NAME=*surface_name*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: select the initial surface region

Restrictions

Surfaces that are associated with cavity radiation are subject to the following restrictions in addition to the general surface definition restrictions outlined in *Element-Based Surface Definition*:

- Surfaces cannot overlap because of the ambiguity that would result in the associated property definitions and in the blocking specification.
- A surface can be used only in one cavity definition (the same surface cannot appear in two different cavities).

In addition, the three-dimensional quadrilateral facets should be as close to planar as possible; otherwise, the quality of the view factor calculations will be compromised.

Controlling Spurious Spatial Oscillations

The radiation flux for each facet is calculated based on the average of the nodal temperatures on that facet (see *Cavity radiation*). This value of radiation flux is then distributed to each node in proportion to its area. Consequently, the mesh must be sufficiently fine that temperature differences across elements are small. Otherwise, computed fluxes at nodes with temperatures above the facet average will be excessively low, and the fluxes at nodes with below-average temperatures will be too high. This tends to induce a spatially oscillatory solution. This effect can be eliminated by reducing the element size in the vicinity of high temperature gradients.

Defining Surface Radiation Properties

Cavity radiation problems are intrinsically nonlinear, due to the dependence of the radiative flux on the fourth power of the facet temperature. Further, nonlinearity can be introduced by describing the emissivity, ϵ, as a function of temperature.

Defining the Emissivity

Emissivity is a dimensionless quantity with a value that is greater than or equal to zero and less than or equal to one. A value of $\epsilon = 0$ corresponds to all radiation being reflected by the surface. A value of $\epsilon = 1$ corresponds to black body radiation, where all radiation is absorbed by the surface. You can define the emissivity, ϵ, of a surface as a function of temperature and other predefined field variables.

You must assign a name to the surface property that defines the emissivity.

Input File Usage: Use both of the following options to define the emissivity of a surface:

**SURFACE PROPERTY*, NAME=*property_name*
**EMISSIVITY*

The **EMISSIVITY* option must appear directly after the **SURFACE PROPERTY* option in the model definition section of the input file.

If black body radiation is being defined ($\epsilon = 1$), the following option can be used in the step definition to improve efficiency:

**RADIATION VIEW FACTOR*, REFLECTION=NO

Abaqus/CAE Usage: Use the following input to define gray body radiation:

Interaction module: **Create Interaction Property**: **Cavity radiation**:
enter the emissivity (ϵ)

You can define the emissivity as a function of temperature and/or field variables.

Use the following input to define black body radiation:

Interaction module: **Create Interaction**: **Cavity radiation**:
Use heat reflection: **No**

Controlling the Accuracy of Temperature-Dependent Emissivity Changes

Abaqus/Standard evaluates the emissivity, ϵ, based on the temperature at the start of each increment and uses that emissivity value throughout the increment. When emissivity is a function of temperature or field variables, you can control the time incrementation for the heat transfer or coupled thermal-electrical step by specifying the maximum allowable emissivity change during an increment, $\Delta\epsilon_{max}$. If this tolerance is exceeded, Abaqus/Standard will cut back the increment size until the maximum change in emissivity is less than the specified value. If you do not specify a value for $\Delta\epsilon_{max}$, a default value of 0.1 is used.

Input File Usage: Use either of the following options:

**HEAT TRANSFER*, MXDEM=$\Delta\epsilon_{max}$
**COUPLED THERMAL-ELECTRICAL*, MXDEM=$\Delta\epsilon_{max}$

Abaqus/CAE Usage: Step module: **Create Step**: **Heat transfer** or **Coupled thermal-electric**: **Incrementation**: **Automatic**: **Max. allowable emissivity change per increment**: $\Delta\epsilon_{max}$

Defining the Stefan-Boltzmann Constant and Value of Absolute Zero

You must define the Stefan-Boltzmann constant, σ, and the value of absolute zero, θ^z; there are no default values for these constants.

Input File Usage: **PHYSICAL CONSTANTS*, STEFAN BOLTZMANN=σ, ABSOLUTE ZERO=θ^z

This option can appear anywhere in the model definition portion of the input file.

Abaqus/CAE Usage: Any module: **Model->Edit Attributes->*model_name***. Enter values for **Absolute zero temperature** and **Stefan-Boltzmann constant**

Constructing a Cavity

You construct cavities as collections of the surfaces defined as described above. Each surface can be used only in one cavity definition. Each cavity must have a unique name; this name is used to specify view factor calculations. The cavity name can also be used to request output.

Setting Surface Properties

By default, a cavity is assumed to consist of surfaces for which surface properties have already been defined. Instead, you may define surface properties as part of the cavity definition.

Input File Usage: Use the following option to construct a cavity:

**CAVITY DEFINITION*, NAME=*cavity_name*, SET PROPERTY
surface name, *surface property name*

By using the SET PROPERTY parameter, you define the surface properties used in the cavity, overriding any property defined as part of the surface option.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: select the surface region. Use the **Properties** table to add or edit surfaces and cavity radiation interaction properties (emissivity).

Creating a Closed Cavity

By default, a cavity is assumed to be closed.

Input File Usage: Use the following option to construct a closed cavity:

**CAVITY DEFINITION*, NAME=*cavity_name*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: **Definition**: **Closed**

Creating an Open Cavity

You can specify an open cavity by defining the reference temperature of the external medium. This ambient temperature value is converted to an absolute temperature scale based on the definition of absolute zero. You can verify the degree of opening in the cavity by specifying a tolerance for the accuracy of the view factor calculations; radiation to the external medium will take place only if the deviation of the sum of the view factors from unity is more than this tolerance. See *Controlling the Accuracy of View Factor Calculations* below for details.

Input File Usage: Use the following option to create an open cavity:

**CAVITY DEFINITION*, NAME=*cavity_name*, AMBIENT TEMP=θ_{amb}

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: **Definition**: **Open**, **Ambient temperature**: θ_{amb}

Creating a Cavity with Multiple Openings or Complex Ambient Conditions

The open cavity definition allows for a cavity with a single opening into an ambient environment with a single, constant temperature value. If the cavity has multiple openings or the ambient temperature is not constant, you should model the surroundings differently.

You should close any cavity openings with elements, and prescribe the temperatures of the external media on these elements. Since the cavity is now closed, you should not specify an ambient temperature with the cavity definition. The temperature definition that you use for the closing elements provides the ambient temperature, and it allows you to specify different temperatures, including variable temperatures, at the cavity openings. The elements modeling the external media should not share nodes with the cavity elements (so that conduction will not take place between them). The surfaces defined by the external media elements should have an emissivity of 1.

Decomposing Large Cavities in Parallel

By default, Abaqus/Standard uses a single working thread for the calculation of the view factor matrix and solution of the radiative heat transfer equations (see *Cavity radiation*). This method is robust and works well for small cavities composed of hundreds of facets, but it becomes inefficient and computationally expensive for large cavities composed of thousand of facets. Moreover, the memory requirements for these cavities may be prohibitively large for a single computational node (the view factor matrix is the size of the number of facets squared). In these cases you should consider allowing Abaqus/Standard to decompose the cavity among all CPUs during view factor calculations and solution of the radiative heat transfer equations.

Input File Usage: Use the following option to activate cavity parallel decomposition:

**CAVITY DEFINITION*, NAME=*cavity_name*, PARALLEL DECOMPOSITION=ON

Abaqus/CAE Usage: Cavity parallel decomposition is not supported in Abaqus/CAE.

Solving Radiative Heat Transfer Equations in Parallel

Abaqus/Standard uses an iterative solution technique for obtaining the radiative heat fluxes when cavity parallel decomposition is enabled. This technique is based on Krylov methods, employs a preconditioner, and uses only MPI-based parallelization (see *Parallel Execution in Abaqus/Standard* for details). This iterative technique is used only to solve the cavity radiation equations and does not require user intervention. You may still opt to use the either the iterative or direct sparse solvers for the solution of the heat transfer finite element equations.

Convergence of Models with Decomposed Cavities

The exact cavity radiation equations are solved whether parallel decomposition is allowed or not; however, when parallel decomposition is active, Abaqus/Standard may require more iterations to obtain a solution. This slower rate of convergence comes from an approximation to the Jacobian (the linearization of the radiation fluxes) that is based on small changes of the irradiation (any part not due to emission from the surface). Models involving surfaces with low emissivities and steady-state analyses might be especially affected. If you encounter convergence problems with parallel decomposed cavities, you may consider

- changing the analysis from steady-state to transient (*Uncoupled Heat Transfer Analysis*); or
- allowing more solver iterations per time increment (*Convergence Criteria for Nonlinear Problems*).

Kinematic Constraints on Models with Decomposed Cavities

Kinematic constraints (for example, coupling constraints, linear constraint equations, multi-point constraints, or surface-based tie constraints) can be applied to any node or surface belonging to a cavity where parallel decomposition is allowed. However, the nodes or surfaces must be the independent (main) nodes or surfaces in the constraint definition.

Defining Cavity Symmetries

Taking advantage of geometric symmetry can reduce computational model size and simulation time. Instead of modeling all of the parts or components in a symmetric assembly, you can model a smaller repeated component and take symmetry into account in the definition of the cavity radiation interaction. In Abaqus/Standard cavity definitions with defined symmetries take into account the radiation interactions between each cavity facet and between all of the facets in the cavity and all of its symmetric images. Abaqus/Standard does not check that the model created using cavity symmetries is physically realistic. You must check the input and results carefully to ensure that a valid model is created.

You must assign a name to each radiation symmetry definition for reference by a radiation view factor definition. The radiation view factor definition and corresponding radiation symmetry definition must appear in the same step.

Cyclic, periodic, and/or reflection symmetries can be defined as described below.

Input File Usage: Use all of the following options to define symmetry in a cavity radiation problem:

**RADIATION VIEW FACTOR*, SYMMETRY=*symmetry_name*
**RADIATION SYMMETRY*, NAME=*symmetry_name*
**REFLECTION* and/or **PERIODIC* and/or **CYCLIC*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: **Symmetry**: **Reflection**, **Periodic**, and/or **Cyclic**

Reflection Symmetry

You define reflection symmetry to create a cavity that is composed of the user-defined cavity surface plus its reflected image through a line or plane. You must identify the dimensionality of the cavity when you define reflection symmetry.

Reflection of Two-Dimensional Cavities

You can define the cavity symmetry by reflecting the cavity surface through a line, as shown in *Figure 1*.

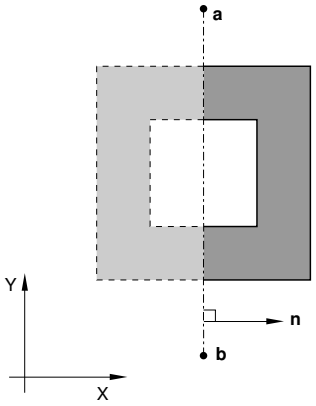


Figure 1: Reflection symmetry through a line.

This type of reflection can be used only with two-dimensional cavities.

Input File Usage: **REFLECTION*, TYPE=LINE

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: **Symmetry**: **Reflection**: select the symmetry line

Reflection of Three-Dimensional Cavities

You can define the cavity symmetry by reflecting the cavity surface through a plane, as shown in *Figure 2*.

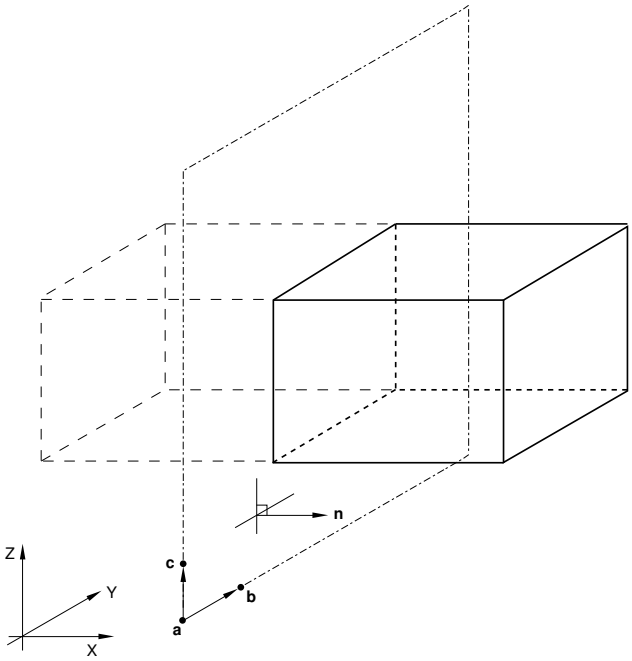


Figure 2: Reflection symmetry through a plane.

This type of reflection can be used only with three-dimensional cavities.

Input File Usage: **REFLECTION*, TYPE=PLANE

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: **Symmetry**: **Reflection**: select the symmetry plane

Reflection of Axisymmetric Cavities

You can define the cavity symmetry by reflecting the cavity surface through a line of constant z-coordinate, as shown in *Figure 3*.

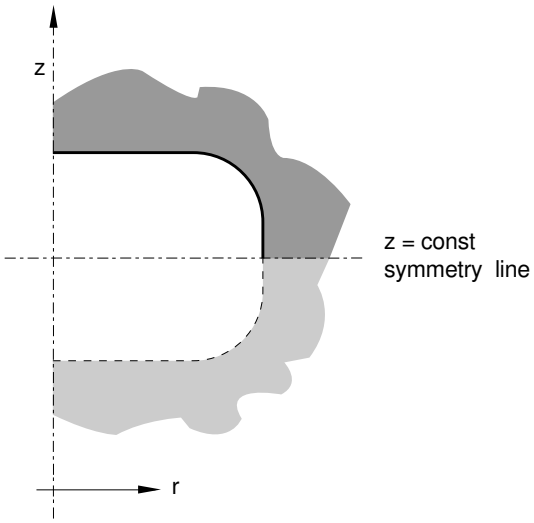


Figure 3: Reflection symmetry through a line of constant z-coordinate.

This type of reflection can be used only with axisymmetric cavities.

Input File Usage: **REFLECTION*, TYPE=ZCONST

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: **Symmetry**: **Reflection**: enter the z-axis symmetry value for the line of symmetry

Periodic Symmetry

You can define cavity symmetry by periodic repetition in a given direction. Physically, periodic symmetry is understood as an infinite number of repetitions of the same image at a periodic interval. Numerically, periodic symmetry has to be represented by a finite number of repetitions of the periodic image. You can define the number of repetitions used in the numerical calculation, n.

The periodic symmetry will result in a cavity composed of the user-defined cavity plus twice n similar images, since the periodic symmetry is assumed to apply in both the positive and negative directions. By default, n=2.

Although symmetries do not increase the size of the view factor matrix, they do make its calculation more expensive. Therefore, the number of repetitions should be minimized, but the value of n should be large enough that the view factor matrix is calculated accurately. Output variable VFTOT can be used to check the amount of closure implied by the symmetry. (See *Controlling the Accuracy of View Factor Calculations* below.) Periodic symmetry for defining the cavity radiation view factor matrix does not impose symmetry conditions automatically in the heat transfer analysis. It may be necessary to impose appropriate constraints on the temperature and loading conditions at the nodes on the periodic symmetry planes to obtain a meaningful solution from the underlying heat transfer analysis.

You must identify the dimensionality of the cavity when you define periodic symmetry.

Periodic Symmetry of Two-Dimensional Cavities

You can create a cavity that is composed of a series of similar images generated by repetition along a two-dimensional distance vector, as shown in *Figure 4*.

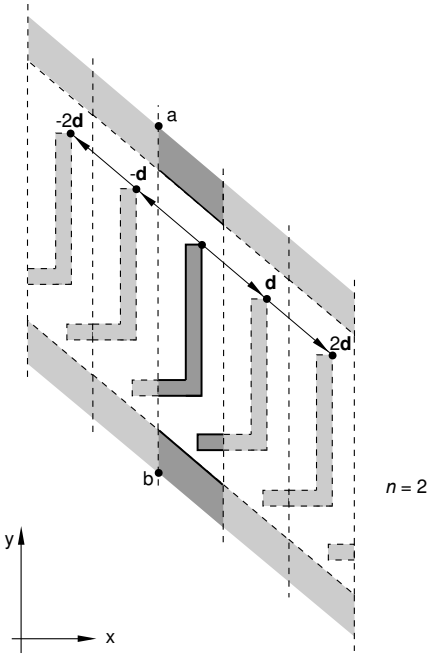


Figure 4: Two-dimensional periodic symmetry.

The repeated images are bounded by lines parallel to line *ab*. The distance vector must be defined so that it points away from line *ab* and into the domain of the model. This type of periodic symmetry can be used only with two-dimensional cavities.

Input File Usage: **PERIODIC*, TYPE=2D, NR=*n*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: **Symmetry**: **Periodic**: **Number of periodic symmetries**: *n*

Periodic Symmetry of Three-Dimensional Cavities

You can create a cavity that is composed of a series of similar images generated by repetition along a three-dimensional distance vector, as shown in *Figure 5*. The repeated images are bounded by planes that are parallel to plane *abc*. The distance vector must be defined so that it points away from plane *abc* and into the domain of the model. This type of periodic symmetry can be used only with three-dimensional cavities.

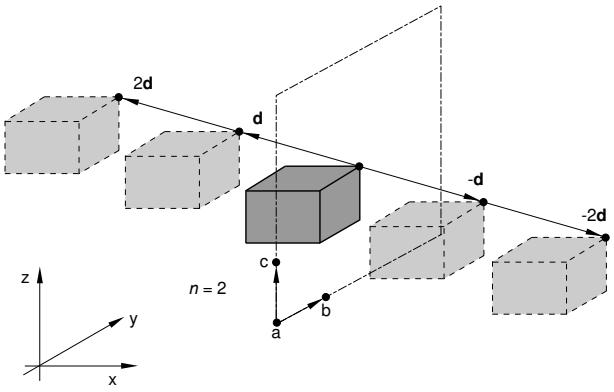


Figure 5: Three-dimensional periodic symmetry.

Input File Usage: **PERIODIC*, TYPE=3D, NR=*n*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: **Symmetry**: **Periodic**: **Number of periodic symmetries**: *n*

Periodic Symmetry of Axisymmetric Cavities

You can create a cavity that is composed of a series of similar images generated by repetition in the *z*-direction, as shown in *Figure 6*.

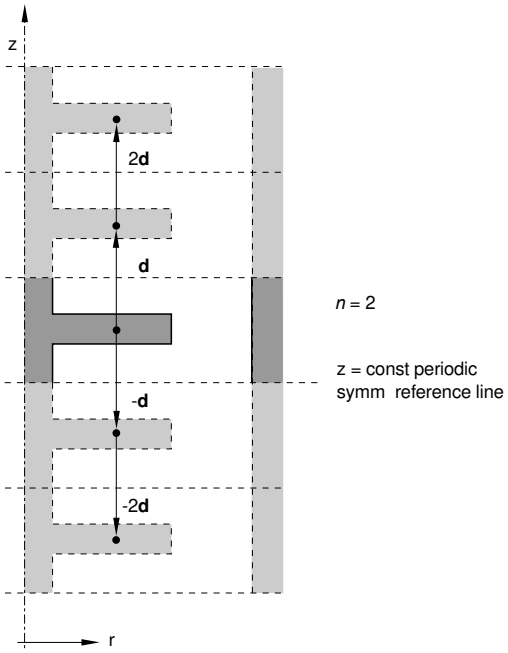


Figure 6: Axisymmetric periodic symmetry.

The repeated images are bounded by lines of constant z-coordinate. The z-distance vector must be defined so that it points away from the z-constant periodic symmetry reference line and into the domain of the model. This type of periodic symmetry can be used only with axisymmetric cavities.

Input File Usage: **PERIODIC*, TYPE=ZDIR, NR=*n*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: **Symmetry**: **Periodic**: **Number of periodic symmetries**: *n*

Cyclic Symmetry

You can define cavity symmetry by cyclic repetition of the user-defined cavity surface about a point or an axis. The cavity defined by cyclic repetition must cover 360°.

You must define the number of cyclically similar images that compose the cavity, n. The angle of rotation about a point or axis used to create cyclically similar images is equal to $360°/n$.

You must identify the dimensionality of the cavity when you define cyclic symmetry.

Cyclic Symmetry of Two-Dimensional Cavities

You can define the cavity symmetry by rotating the cavity about a point, l, as shown in *Figure 7*.

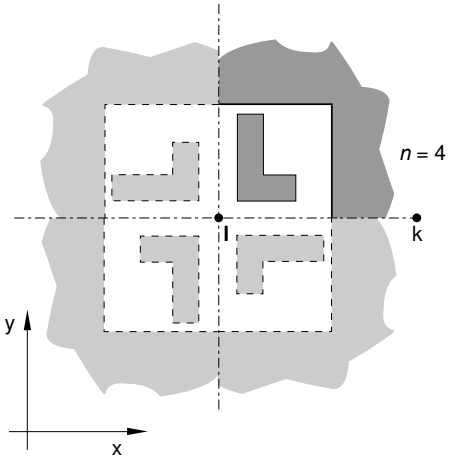


Figure 7: Cyclic symmetry about a point.

The cavity surface defined in the model must be bounded by the line *lk* and a line passing through *l* at an angle, measured counterclockwise when looking into the plane of the model, of 360°/*n* to *lk*. This type of cyclic symmetry can be used only for two-dimensional cavities.

Input File Usage: **CYCLIC*, TYPE=POINT, NC=*n*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**:
Symmetry: **Cyclic**: toggle on **Use cyclic symmetric**,
Total number of sectors: *n*

Cyclic Symmetry of Three-Dimensional Cavities

You can define the cavity symmetry by rotating the cavity about an axis, *lm*, as shown in *Figure 8*. The cavity surface defined in the model must be bounded by the plane *lmk* and a plane passing through the line *lm* at an angle, measured clockwise when looking from *l* to *m*, of 360°/*n* to *lmk*. Line *lk* must be normal to line *lm*. This type of cyclic symmetry can be used only for three-dimensional cavities.

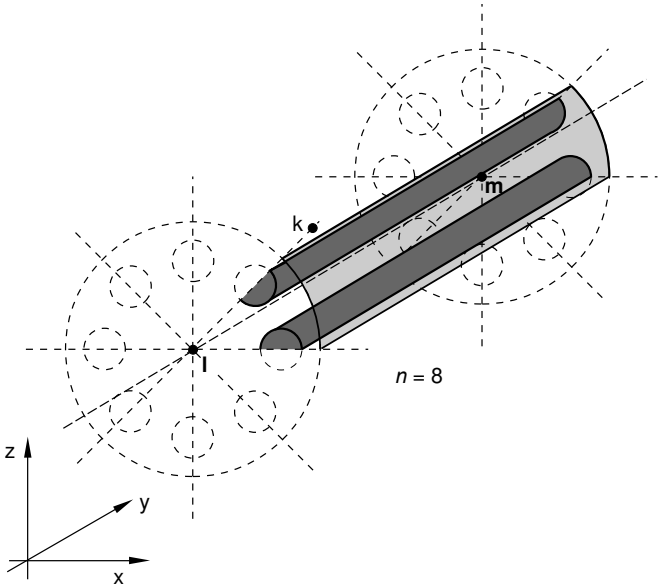


Figure 8: Cyclic symmetry about an axis.

Input File Usage: **CYCLIC*, TYPE=AXIS, NC=*n*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: **Symmetry**: **Cyclic**: toggle on **Use cyclic symmetric**,
Total number of sectors: *n*

Combining Symmetries

Reflection, periodic, and cyclic symmetries can be combined as shown in *Table 1*. *Figure 9* through *Figure 12* illustrate some possible symmetry combinations.

Table 1: Permissible number of symmetry definitions used in combination.

Reflection	Periodic	Cyclic	2D	3D	Axi	Restrictions
1	0	0	•	•	•	
2	0	0	•	•		$\mathbf{n}_1 \perp \mathbf{n}_2$
3	0	0		•		$\mathbf{n}_1 \perp \mathbf{n}_2 \perp \mathbf{n}_3 \perp \mathbf{n}_1$
0	1	0	•	•	•	
0	2	0	•	•		
0	3	0		•		
1	1	0	•	•		$\mathbf{n} \perp \mathbf{d}$
1	2	0		•		$\mathbf{d}_1 \perp \mathbf{n} \perp \mathbf{d}_2$
2	1	0		•		$\mathbf{d} \perp \mathbf{n}_1 \perp \mathbf{n}_2 \perp \mathbf{d}$
0	0	1	•	•		

Reflection	Periodic	Cyclic	2D	3D	Axi	Restrictions
1	0	1		•		$\mathbf{n} \parallel \mathbf{l} - \mathbf{m}$
0	1	1		•		$\mathbf{d} \parallel \mathbf{l} - \mathbf{m}$

$\mathbf{n}$, $\mathbf{n_1}$, $\mathbf{n_2}$, $\mathbf{n_3}$ are normals to lines or planes of reflection symmetry.

$\mathbf{d}$, $\mathbf{d_1}$, $\mathbf{d_2}$ are distance vectors used to define periodic symmetry.

$\mathbf{l} - \mathbf{m}$ is the direction of the axis of cyclic symmetry in three-dimensional cases.

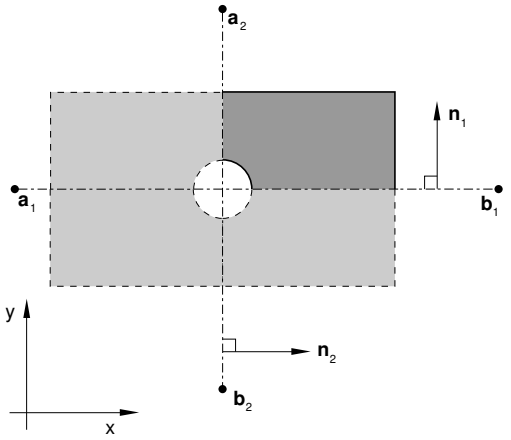


Figure 9: Combination of two reflection symmetries in two dimensions.

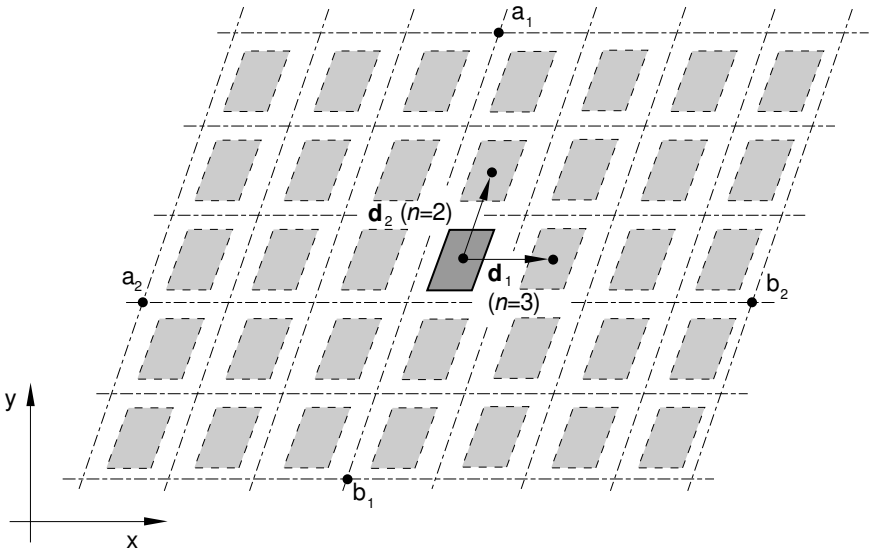


Figure 10: Combination of two periodic symmetries in two dimensions.

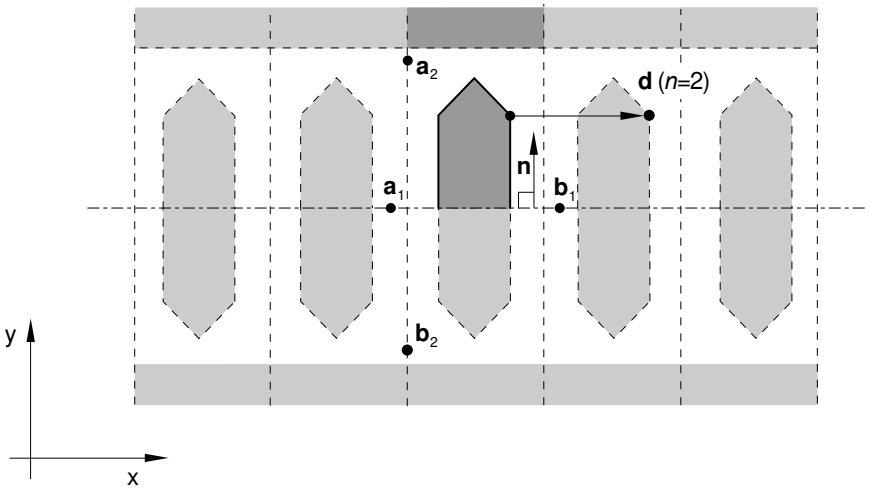


Figure 11: Combination of one reflection symmetry and one periodic symmetry in two dimensions.

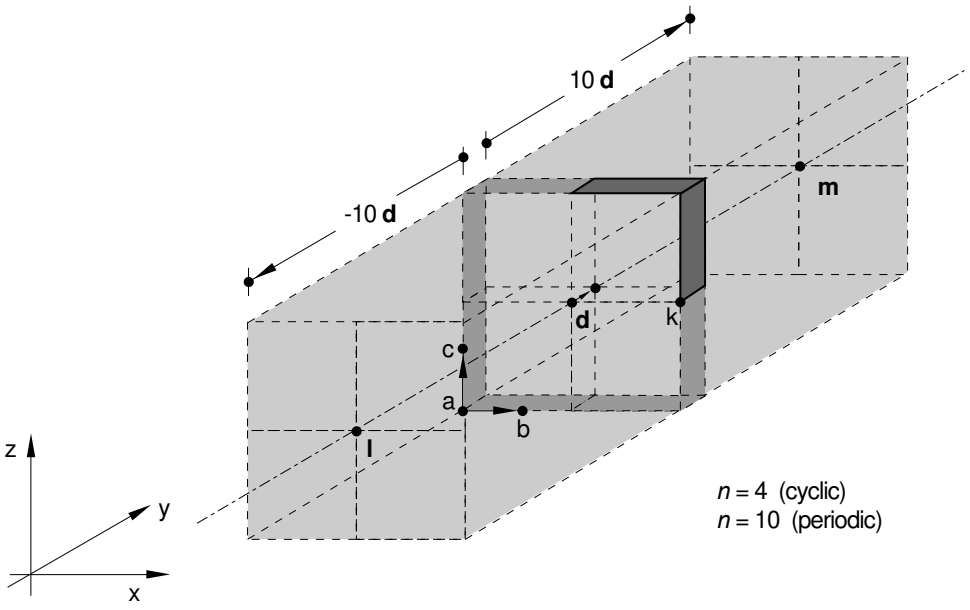


Figure 12: Combination of one cyclic symmetry and one periodic symmetry in three dimensions.

Prescribing Motion during a Cavity Radiation Analysis

In many cavity radiation problems such as simulations of manufacturing sequences, radiation view factors change because surfaces are moved during the analysis. You can specify surface motions during heat transfer or coupled thermal-electrical analysis.

The prescribed motions affect only the calculation of view factors (and, therefore, radiation fluxes) in heat transfer due to cavity radiation. They do not affect heat conduction, storage, or distributed flux contributions.

You can define both the translational and rotational components of the motion within a step independently. For example, you can prescribe the translational motion of a node set according to a certain amplitude function and

then prescribe the rotational motion of the node set according to a different amplitude function. In each step, each component of motion can be specified only once for any particular node.

Motions can also be prescribed during steps in which the cavity radiation is turned off, as described below.

Translational Motion

Translations, $\mathbf{u}^t$, are specified in terms of global x-, y-, and z-components unless a local coordinate system is defined at the nodes for which motion is specified; then translations are specified in terms of local x-, y-, and z-components (see *Transformed Coordinate Systems*).

Translational displacements are always specified as total values of translational motion. This treatment of translations is consistent with that used for displacement boundary conditions (*Boundary Conditions*) in stress/displacement analyses. The default is to apply translational motion.

Translational velocities can also be specified. Translational velocities always refer to the current step; therefore, the rate of translational motion specified as a velocity is in effect only during the step for which it is defined. This behavior is different from velocity boundary conditions, where velocities stay in effect in subsequent steps if they are not redefined.

Input File Usage: Use either of the following options to prescribe translational motion:

**MOTION*, TRANSLATION, TYPE=DISPLACEMENT
**MOTION*, TRANSLATION, TYPE=VELOCITY

Abaqus/CAE Usage: Surface motion is not supported with cavity radiation in Abaqus/CAE.

Rotational Motion

Displacements due to a rigid body rotation, $\mathbf{u}^r$, can be defined by specifying the magnitude of the rotation and the rotation axis. In three dimensions the rotation axis is defined by specifying two points, $\mathbf{a}$ and $\mathbf{b}$, on the axis of rotation. In two dimensions the rotation axis is assumed to be normal to the plane of the model and is defined by specifying one point, $\mathbf{a}$.

The coordinates of the points defining the axis of rotation must be defined in the configuration at the beginning of the step for which rigid body rotation is being defined.

Motion due to rigid body rotation during a step is specified as the amount of rotation that takes place during that step only. Therefore, the rigid body rotation specified during a step is local to that step; if no rigid body rotation is specified in the following step, no further rotation occurs.

The treatment of rigid body rotations is different from that of translations: rigid body rotations are specified incrementally from step to step while translations are specified as total values.

Input File Usage: Use either of the following options to prescribe rotational motion:

**MOTION*, ROTATION, TYPE=DISPLACEMENT
**MOTION*, ROTATION, TYPE=VELOCITY

Abaqus/CAE Usage: Surface motion is not supported with cavity radiation in Abaqus/CAE.

Prescribing Large Rotational Motions

Prescribed rotational motions of more than 2π radians or complex sequences of rotations about different directions in three-dimensional models are most simply defined by specifying rotational velocities, which allows the definition to be given in terms of the angular velocity instead of the total rotation. Abaqus/Standard calculates the increment of rotation as the average of the angular velocities at the beginning and end of each increment multiplied by the time increment. (See *Conventions*.)

Example

For example, if a rotation of 6π about the z-axis is required, with no rotation about the x- and y-axes, and assuming a step time of 1.0, specify a constant angular velocity of 6π as follows:

```
*MOTION, TYPE=VELOCITY, ROTATION
 node (node set), 18.84955592, 0., 0., 0., 0., 0., 1.
```

The angular velocity will be constant since the default variation for motions prescribed using a predefined velocity field in a heat transfer or coupled thermal-electrical step (both steady-state and transient) is a step function (see *Defining an Analysis*). An amplitude reference could be used to specify other variations of the angular velocity.

If, in the next step, the same node (or node set) should have an additional rotation of $\pi/2$ radians about the global x-axis, assuming again a step time of 1.0, prescribe a constant angular velocity as follows:

```
*MOTION, TYPE=VELOCITY, ROTATION
 node (node set), 1.570796327, 0., 0., 0., 1., 0., 0.
```

Prescribing Simultaneous Rigid Body Rotations

Motions involving two or more simultaneous rigid body rotations about different axes cannot be specified directly. An example of simultaneous rigid body rotations is a satellite rotating about its own axis while orbiting the earth. Such complex motions can be defined with user subroutine *UMOTION*. This subroutine allows specification of the time variation of the magnitude of the translational components of the motion (degrees of freedom 1–3) at each node.

If you specify the magnitude of the translation as part of the prescribed motion definition, it will be modified by the amplitude curve (if any) and passed into subroutine *UMOTION*, where it can be redefined.

When user subroutine *UMOTION* is used to define the motion of a certain node set in a step, only one prescribed motion can be defined in that step for that node set. The complete motion of all nodes in the node set during the step must be defined in the user subroutine.

Input File Usage: **MOTION*, USER

Abaqus/CAE Usage: Surface motion is not supported with cavity radiation in Abaqus/CAE.

Simultaneous Translational and Rotational Motion

Whenever simultaneous translational and rotational motion is specified, the total motion of a node during step k is defined as

$$\mathbf{x} = \mathbf{X} + \mathbf{u}^t + \sum_{i=1}^{k} \mathbf{u}_i^r,$$

where $\mathbf{x}(t)$ is the current location of the node due to the specified motion history, $\mathbf{X}$ is the original location of the node, $\mathbf{u}^t(t)$ is the displacement of the node due to the translational motion specified in the step, and $\mathbf{u}_i^r$ is the displacement of the node due to rigid body rotation during step i.

In these cases the translation is applied first and the rotation is then assumed to be about the translated (material) axis. In other words, the displacement $\mathbf{u}_i^r$ due to rigid body rotation during step i is computed as the rotation about an axis defined by points $\mathbf{a}_i^t$ and $\mathbf{b}_i^t$ where

$$\mathbf{a}_i^t = \mathbf{a} + \mathbf{u}_i^t, \qquad \mathbf{b}_i^t = \mathbf{b} + \mathbf{u}_i^t.$$

In the preceding equations $\mathbf{a}$ and $\mathbf{b}$ are the locations of the points used to define the axis of rotation for the prescribed rotational motion (they refer to the configuration at the beginning of step i) and $\mathbf{u}_i^t(t)$ is the displacement due to translational motion during the step ($\mathbf{u}_i^t(t) = \mathbf{u}^t(t) - \mathbf{u}_{i-1}^t(T_{i-1})$, where T_{i-1} is the time at the end of step $i-1$).

Example

As an example, consider a three-dimensional problem with x–y planar motion as shown in *Figure 13*.

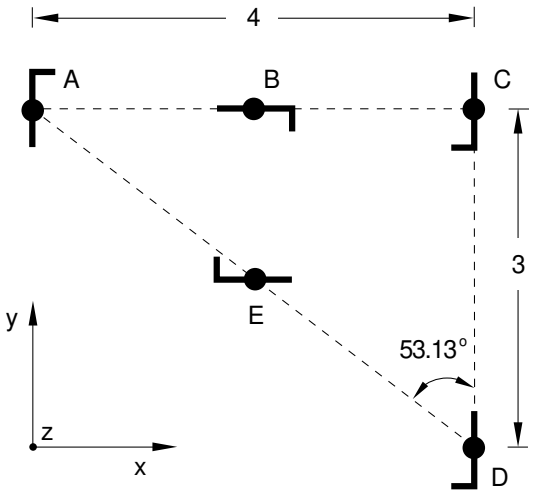


Figure 13: Planar motion example.

The centroid of the object of interest is initially located at $x = 0, y = 3, z = 0$. In the first step the object is translated 4 length units in the x-direction while at the same time it rotates clockwise 180° (π radians) about the z-axis at constant angular velocity. This motion moves the object from position A to position C in *Figure 13*. Halfway through this motion, at position B, the displacements due to the rigid body rotation are calculated by applying the translation to the z-axis (the axis of rotation) and then applying a 90° rotation about this translated axis.

In the second step the object is translated −3 length units in the y-direction only. This motion places the object at position D with no additional rotation. Finally, in the third step the object is simultaneously translated 5 length units at an angle of 53.13° to the y-direction and rotated clockwise, again at constant angular velocity, through 180° about the z-axis. This motion returns the object to its original position.

Assuming that each step time is 1.0, the input required for the above motion sequence is as follows:

First step:

```
*MOTION
 node set, 1, 1, 4.
*MOTION, ROTATION, TYPE=VELOCITY
 node set, 3.14159265, 0., 3., 0., 0., 3., -1.
```

Second step:

```
*MOTION
 node set, 2, 2, -3.
```

Third step:

```
*MOTION
 node set, 1, 2, 0.
*MOTION, ROTATION, TYPE=VELOCITY
 node set, 3.14159265, 4., 0., 0., 4., 0., -1.
```

Controlling the Time Variation of the Motion

For any prescribed motion you can refer to an amplitude curve that gives the time variation of the motion throughout a step (see *Amplitude Curves*).

Input File Usage: Use both of the following options:

**AMPLITUDE*, NAME=*amplitude*
**MOTION*, AMPLITUDE=*amplitude*

Abaqus/CAE Usage: Surface motion is not supported with cavity radiation in Abaqus/CAE.

Controlling the Frequency of View Factor Recalculation due to Motion

You can control how view factors are recalculated during a step as a result of prescribed motion by specifying a value for the maximum allowable motion, *max*, for a particular node set. View factor recalculation is triggered if a displacement component at any node in the specified node set exceeds the specified value for *max*.

You must respecify the value of *max* and the node set in every step where recalculation is required; the values do not remain in effect for subsequent steps.

View factor recalculation can be expensive; use discretion when choosing a value for *max*.

Input File Usage: **RADIATION VIEW FACTOR*, MDISP=*max*, NSET=*nset*

The *max* and *nset* values must always be specified together.

Abaqus/CAE Usage: View factor recalculation due to motion is not supported with cavity radiation in Abaqus/CAE.

Controlling View Factor Calculation during the Analysis

The cavity radiation capability can be used in applications such as the simulation of manufacturing sequences where radiation view factors change during the simulation. Therefore, radiation view factor definitions provide significant flexibility for the control of view factor calculations during a step.

Multiple radiation view factor definitions can be specified within a step definition if different types of radiation and view factor calculations are required for different cavities. Different types of view factor calculations can be specified for the same cavity in different steps of the analysis.

By default, view factors are calculated at the beginning of the first step that includes a radiation view factor definition. View factors are recalculated at the beginning of a subsequent step only if the view factor definition changes in that step; for example, if different surface blocking checks are specified for the same cavity. In a restart analysis Abaqus/Standard reads the radiation view factors from the user-specified restart step and increment and recalculates the view factors only if the view factor definitions have changed.

You can specify the name of the cavity for which radiation view factor control is being specified. If you do not specify a cavity name, the radiation view factor definition applies to all cavities in the model.

Input File Usage: **RADIATION VIEW FACTOR*, CAVITY=*cavity_name*

Abaqus/CAE Usage: Radiation view factors are defined separately for each cavity radiation interaction and apply to all steps in which that interaction is active.

Activating and Deactivating Cavity Radiation

There are practical situations in which it may be useful to switch cavity radiation effects on and off during the analysis. For example, radiation may be taking place in a cavity that is then filled with a fluid so that radiation is no longer significant; later in the analysis, radiation may resume when the fluid is drained from the cavity. In such cases you can use a radiation view factor definition to switch the radiation on and off in any particular cavity during one or more steps of the analysis.

When cavity radiation is switched back on after having been switched off, Abaqus/Standard will use the last view factors calculated in the last step in which cavity radiation was active. However, if motion is prescribed during the time that the cavity radiation is switched off and one of the displacement components of a node in the specified node set exceeds the value for the maximum allowable motion, *max*, specified in the step during which cavity radiation is switched off, the view factors will be recalculated at the beginning of the step in which the cavity radiation is switched back on.

Input File Usage: Use the following option to turn view factor calculation off for a step:

**RADIATION VIEW FACTOR*, OFF

Use one of the following options to turn view factor calculation back on in a subsequent step:

**RADIATION VIEW FACTOR*
**RADIATION VIEW FACTOR*, MDISP=*max*, NSET=*nset*

Abaqus/CAE Usage: Radiation view factors cannot be turned off or on for a selected step. You can use the following options to turn a cavity radiation interaction off or on:

Interaction module: **Interaction Manager**: select a step and a cavity radiation interaction, **Activate** or **Deactivate**

Controlling the Accuracy of View Factor Calculations

Abaqus/Standard uses a progressive integration scheme for view factor calculation. When facets are sufficiently far from each other, a lumped area approximation is used. If the facets are close to each other but one of the facets is much larger than the other, an infinitesimal-to-finite approximation is used. For all other cases a contour integral is numerically calculated to compute the view factor. See *View factor calculation* for details.

Two nondimensional parameters are calculated for each facet pair to determine which integration scheme is used:

$$r_1 = \frac{d^2}{A_{\max}} \qquad \text{and} \qquad r_2 = \frac{A_{\max}}{A_{\min}},$$

where $A_{\min}$ is the area of the smaller facet, $A_{\max}$ is the area of the larger facet, and d is the distance between their centroids. The lumped area approximation is used whenever the nondimensional distance square parameter $r_1 > P_1$, where P_1 has a default value of 5.0. If $r_1 \le P_1$, an infinitesimal-to-finite area approximation is used if the facet area ratio $r_2 > P_2$, where P_2 has a default value of 64.0. Otherwise, a more precise calculation is performed, involving the numerical integration of a contour integral.

You can customize the accuracy and speed of the view factor calculation by specifying the parameters P_1 and P_2 and the number of integration points per edge. For example, Abaqus/Standard will used lumped area approximations throughout the whole model if P_1 is set to zero. Likewise, the more precise, albeit more expensive, numerical integration method will always be used if P_1 and P_2 are set to very large numbers.

Input File Usage: **RADIATION VIEW FACTOR*, LUMPED AREA=*P1*, INFINITESIMAL=*P2*, INTEGRATION=*integration points per edge*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: **View factors**: enter new values or accept the defaults for **Infinitesimal facet area ratio**, **Gauss integration points per edge**, and **Lumped area distance-square value**

View Factor Calculation Checks for Closed Cavities

You can provide a tolerance on the accuracy of the view factor calculation. In a closed cavity the sum of the view factors for each cavity facet should be one. Abaqus/Standard compares the value of the specified tolerance to the largest view factor matrix row sum deviation from unity; that is, $\max_j(|\sum_i F_{ij} - 1|)$. If the tolerance is violated for a closed cavity, the analysis is terminated. The default view factor tolerance is 0.05. Failure to meet this criterion may indicate a need for mesh refinement.

Input File Usage: **RADIATION VIEW FACTOR*, VTOL=*tolerance*

Abaqus/CAE Usage: Interaction module: **Create interaction**: **Cavity radiation**: **View factors**: **Accuracy tolerance**: *tolerance*

View Factor Calculations in Cavities with Symmetries

The view factor calculations account for the closure of a cavity implied by any cavity symmetries. For cavities without periodic or cyclic symmetries the view factors are calculated exactly for two-dimensional geometries, but approximations are made for axisymmetric and three-dimensional geometries. These approximations become less accurate as the distance between surfaces decreases. Define heat radiation to model closely spaced surfaces (see *Thermal Contact Properties*).

View Factor Calculations in Open Cavities

If the sum of the view factors for any facet in an open cavity (defined by specifying a value for the ambient temperature) deviates from unity by more than the specified view factor tolerance, radiation to the ambience will take place from all facets. In nearly closed cavities this deviation may be small. If the tolerance is not violated, radiation to the external medium is not included even though the cavity is defined to be open; a warning message is issued to this effect. You can reduce the view factor tolerance to include such radiation.

Controlling Checks for Surface Blocking

Heat is transferred between surfaces that have unobstructed direct views of each other (see *Figure 14*); "blocking" may occur in geometrically complex cavities.

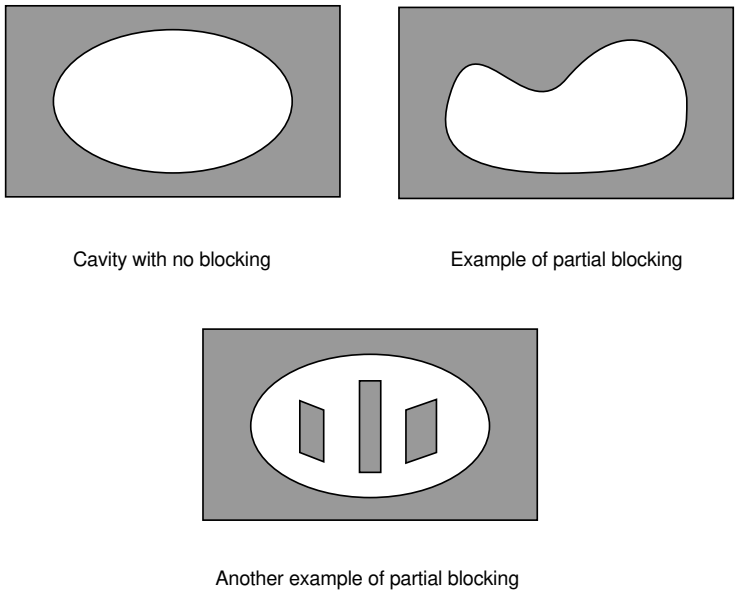


Figure 14: Illustrations of blocking.

Surface blocking checks may be computationally expensive in cavities with many surfaces; therefore, significant computational time may be saved by specifying which surfaces are potential blocking surfaces, as described below.

View factor calculations with blocking surfaces are especially sensitive to mesh refinement. If a mesh is too coarse, the view factors may not add up to one (in a closed cavity). To obtain accurate results, the mesh should be refined until the view factors can be summed accurately.

Full Blocking Checks

By default, Abaqus/Standard will check for blocking of every surface with itself and all other surfaces.

Input File Usage: **RADIATION VIEW FACTOR*, BLOCKING=ALL

Abaqus/CAE Usage: Interaction module: **Create interaction**: **Cavity radiation**: **Properties**: **Blocking surface checks**: **All**

Partial Blocking Checks

You can specify a list of the potential blocking surfaces in the cavity.

Input File Usage: **RADIATION VIEW FACTOR*, BLOCKING=PARTIAL

Abaqus/CAE Usage: Interaction module: **Create interaction**: **Cavity radiation**: **Properties**: **Blocking surface checks**: **Partial**

No Blocking Checks

You can indicate that there are no blocking surfaces in the cavity; in this case Abaqus omits all checks for blocking.

Input File Usage: **RADIATION VIEW FACTOR*, BLOCKING=NO

Abaqus/CAE Usage: Interaction module: **Create interaction**: **Cavity radiation**: **Properties**: **Blocking surface checks**: **None**

Reducing Computations for Surfaces That Are Far Apart

In cases where there are many surfaces in the cavity, surfaces separated by more than a certain distance may not be able to "see" each other for the purposes of radiation because of blocking by other surfaces. You can specify the distance beyond which view factors need not be calculated, which reduces the computational effort required for the view factor calculations.

Input File Usage: **RADIATION VIEW FACTOR*, RANGE=*distance*

Abaqus/CAE Usage: Interaction module: **Create interaction**: **Cavity radiation**: **View factors**: toggle on **Specify blocking range**: *distance*

Memory Usage in Cavity Radiation Analyses

The cavity radiation heat transfer between facets of a surface in Abaqus is modeled using a full, unsymmetric matrix defining interactions between each node and all others in the cavity. For surfaces with large numbers of nodes this matrix may be large, resulting in memory requirements that are significantly larger than those for the finite element portion of the analysis without the cavity radiation interaction.

To minimize memory requirements and computational cost for cavity radiation heat transfer analysis, the cavity can be defined using a coarser mesh of heat transfer shell elements having a single degree of freedom per node. The overlaid element should have minimal heat capacity and conduction, and it should be used for the definition of the cavity in place of the physical, multiple-degree-of-freedom shell. The overlaid element should be used to define the main surface in a tied coupling constraint (*Mesh Tie Constraints*); the multiple-degree-of-freedom, physical, heat transfer shell element forms the secondary surface.

Initial Conditions

By default, the initial temperature of all nodes is zero. You can specify nonzero initial temperatures in a cavity radiation analysis; see *Defining Initial Temperatures*.

In a heat transfer analysis involving forced convection through the mesh, you can define nonzero initial mass flow rates at the nodes of the forced convection/diffusion heat transfer elements in the model (see *Uncoupled Heat Transfer Analysis*).

Boundary Conditions

You can specify boundary conditions to prescribe temperatures (degree of freedom 11) at the nodes (see *Boundary Conditions*). Shell elements have additional temperature degrees of freedom 12, 13, etc. through the thickness (see *Conventions*). Boundary conditions can be specified as functions of time by referring to amplitude curves (*Amplitude Curves*).

For purely diffusive elements, a boundary without any prescribed boundary conditions (natural boundary condition) corresponds to an insulated surface. For forced convection/diffusion elements, only the flux associated with conduction is zero; energy is free to convect across an unloaded surface. This natural boundary condition correctly models areas where fluid is crossing a surface (as, for example, at the upstream and downstream boundaries of the mesh) and prevents spurious reflections of energy back into the mesh.

Loads

The following types of loading can be prescribed in addition to the cavity radiation, as described in *Thermal Loads*:

- Concentrated heat fluxes
- Body fluxes and distributed surface fluxes
- Convective film conditions and radiation conditions

Predefined Fields

You cannot specify temperatures as field variables in heat transfer or coupled thermal-electrical analyses. Boundary conditions should be used instead, as described above.

You can specify values of other user-defined field variables during the analysis. These values will affect field-variable-dependent material properties, if any. See *Predefined Fields*.

Material Options

You must define the radiation properties of the surfaces as described above in *Defining Surface Radiation Properties*. Other thermal properties such as conductivity, density, specific heat, and latent heat are defined as in uncoupled heat transfer analysis—see *Uncoupled Heat Transfer Analysis* and *About Thermal Properties*.

You can specify internal heat generation—see *Internal Heat Generation*.

Thermal expansion coefficients are not meaningful in cavity radiation heat transfer analysis since deformation of the structure is not considered.

Elements

Any of the heat transfer or coupled thermal-electrical elements in Abaqus/Standard can be used in a cavity radiation analysis, including forced convection/diffusion heat transfer elements (see *Choosing the Appropriate Element for an Analysis Type*, *Uncoupled Heat Transfer Analysis*, and *Coupled Thermal-Electrical Analysis*). Coupled temperature-displacement and coupled thermal-electrical-structural elements cannot be used in a cavity radiation analysis.

In addition to the elements that you define, Abaqus/Standard uses internal elements that are generated automatically from your definition of radiation cavities.

Output

The following output variables are available for cavity radiation:

Surface variables

RADFL	Radiation flux per unit area. This variable does include heat flux to ambient in an open cavity.
RADFLA	Radiation flux over a facet.
RADTL	Time integrated radiation per unit area.
RADTLA	Time integrated radiation over a facet.
VFTOT	Total view factor for a facet (sum of the view factor values in the row of the view factor matrix corresponding to the facet).
FTEMP	Facet temperature.

All of the output variables are listed in *Abaqus/Standard Output Variable Identifiers*. Abaqus/CAE supports motion display and can display surface- and element-based results.

Writing the View Factor Matrices to the Results File

You can write the view factor matrices for cavity radiation interactions in heat transfer or coupled thermal-electrical analyses to the results (`.fil`) file if parallel decomposition for the cavity is not enabled.. The entire radiation view factor matrix is written for each cavity radiation element in the specified cavity.

You can control the frequency of view factor matrix output by specifying the required output frequency in increments. The default output frequency is 1. Specify an output frequency of 0 to suppress output. The output will always be written at the last increment of each step unless you specify an output frequency of 0.

The record formats for the results file are described in *Results File*. The file can be written in binary or ASCII format (see *Controlling the Format of the Results File in Abaqus/Standard*).

Input File Usage: **VIEW FACTOR OUTPUT*, CAVITY=*cavity_name*, FREQUENCY=*n*

Abaqus/CAE Usage: View factor output is not supported in Abaqus/CAE.

Requesting Surface Variable Output

For the cavity radiation interaction, you can request cavity-, element-, or surface-based radiation output such as radiation fluxes, view factor totals for a facet, and facet temperatures to the data, results, and/or output database files. The output requests can be repeated as often as necessary to request output for different variables, different cavities, different surfaces, different element sets, etc. The surface variables that can be requested are listed above.

You can specify the particular cavity, element set, or surface for which output is being requested. If you do not specify a cavity, element set, or surface, output will be provided for all cavities in the model. The same cavity, element set, or surface can appear in several radiation output requests.

By default, no cavity radiation data output will be provided. If you define a radiation output request without specifying the desired output variables, all six cavity radiation surface variables will be output.

You can control the frequency of radiation output by specifying the required output frequency in increments. The default output frequency is 1. Specify an output frequency of 0 to suppress output. The output will always be written at the last increment of each step unless you specify an output frequency of 0.

Input File Usage: Use one of the following options to obtain output in the data file:

**RADIATION PRINT*, CAVITY=*cavity_name*, FREQUENCY=*n*
**RADIATION PRINT*, ELSET=*element_set*, FREQUENCY=*n*
**RADIATION PRINT*, SURFACE=*surface_name*, FREQUENCY=*n*

Use one of the following options to obtain output in the results file:

**RADIATION FILE*, CAVITY=*cavity_name*, FREQUENCY=*n*
**RADIATION FILE*, ELSET=*element_set*, FREQUENCY=*n*
**RADIATION FILE*, SURFACE=*surface_name*, FREQUENCY=*n*

Use the first option and one of the subsequent options to obtain output in the output database:

**OUTPUT*, FREQUENCY=*n*
**RADIATION OUTPUT*, CAVITY=*cavity_name*
**RADIATION OUTPUT*, ELSET=*element_set*
**RADIATION OUTPUT*, SURFACE=*surface_name*

Abaqus/CAE Usage: Cavity radiation output to the data file and the results file are not supported in Abaqus/CAE.

Use the following options to obtain output in the output database:

Step module: history output request editor: **Thermal**: select the output variables

Printed Output

The output tables generated by a radiation output request to the data file are organized on a surface-by-surface basis. The rows that will appear in a particular table are defined by choosing a cavity, surface, or element set: each row of a table corresponds to an individual element face that is part of the cavity, surface, or element set chosen. If all of the variables in a row of a table are zero, the row is not printed.

The first column of each table is the element number, and the second column is the element face identifier. You choose the variables to appear in the remaining columns. There is no limit to the number of tables that can be defined.

As an example, consider a heat transfer model containing a cavity named `CAV1`, which, in turn, is composed of surfaces `SURF1` and `SURF2`. If you request output of radiation flux (RADFL) and facet temperature (FTEMP) to the data file for this model, two tables will appear in the data file. One table will contain RADFL and FTEMP output for all element faces composing surface `SURF1`, and the other table will contain the same output variables for all element faces making up surface `SURF2`.

By default, Abaqus/Standard writes a summary of the maximum and minimum values in each column of the table. You can choose to suppress this summary. In addition, you can choose to print the total of each column in the table, which is useful, for example, to sum radiation fluxes over all facets composing a radiation surface. By default, these totals are not printed.

Input File Usage: Use the following option to control output of the summary information to the data file:

**RADIATION PRINT*, SUMMARY=YES *or* NO

Use the following option to control output of the totals to the data file:

**RADIATION PRINT*, TOTALS=YES *or* NO

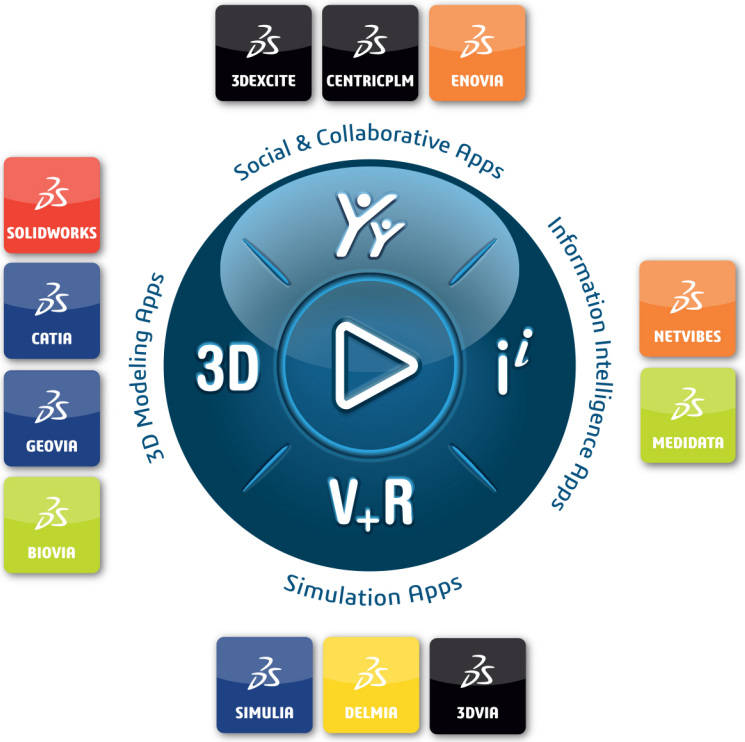
Abaqus/CAE Usage: Cavity radiation output to the data file is not supported in Abaqus/CAE.

Input File Template

The following template shows the options required for a transient, cavity radiation analysis of a closed two-dimensional symmetric cavity. All surfaces within the cavity `topcav` have the same emissivity. The surface `surf2` moves (translation only) during the analysis. In the second step surface `surf2` stops moving, cavity radiation is turned off, all thermal loads except the surface convection are removed, and a steady-state heat transfer analysis is conducted to determine the final temperature of the system.

```
*HEADING
…
*PHYSICAL CONSTANTS, ABSOLUTE ZERO=θ^z, STEFAN BOLTZMANN=σ
*SURFACE, NAME=surf1, PROPERTY=surfp
```

```
 elset1, S1
 elset2, S2
*SURFACE, NAME=surf2, PROPERTY=surfp
 elset3,
*SURFACE PROPERTY, NAME=surfp
*EMISSIVITY
```
Data lines to define the emissivity of the surfaces in the model
```
*CAVITY DEFINITION, NAME=topcav
 surf1, surf2
*INITIAL CONDITIONS, TYPE=TEMPERATURE
```
Data lines to prescribe initial temperatures at the nodes
```
*AMPLITUDE, NAME=motion
```
Data lines to define amplitude curve to be used for motion of surface surf2
```
*AMPLITUDE, NAME=film
```
Data lines to define amplitude curve to be used for the convection film coefficient, h
```
*************
** Step 1
*************
*STEP
```
HEAT TRANSFER, MXDEM=$\Delta\epsilon_{max}$, DELTMX=$\Delta\theta_{max}$
Data line to define incrementation
```
*RADIATION VIEW FACTOR, CAVITY=topcav, VTOL=tol, SYMMETRY=outer,
 NSET=nset, MDISP=max
*RADIATION SYMMETRY, NAME=outer
*REFLECTION, TYPE=LINE
```
Data line to define line of symmetry
```
*MOTION, TRANSLATION, TYPE=DISPLACEMENT, AMPLITUDE=motion
```
Data line to define motion of nodes on surface surf2
```
*CFLUX and/or *DFLUX
```
Data lines to define concentrated and/or distributed fluxes
```
*BOUNDARY
```
Data lines to prescribe temperatures at selected nodes
```
*FILM, FILM AMPLITUDE=film
```
Data lines to define surface convection
```
**
*RADIATION PRINT, CAVITY=topcav, SUMMARY=YES, TOTALS=YES
```
Data lines requesting cavity radiation surface variable output
```
*RADIATION FILE, CAVITY=topcav, FREQUENCY=4
```
Data lines requesting cavity radiation surface variable output
```
*NODE PRINT
```
Data lines requesting nodal output such as temperatures
```
*EL PRINT
```
Data lines requesting element output such as heat flux
```
*END STEP
*************
** Step 2
*************
*STEP
*HEAT TRANSFER, STEADY STATE
```
Data line to define incrementation
```
*RADIATION VIEW FACTOR, OFF
*CFLUX, OP=NEW
*DFLUX, OP=NEW
*END STEP
```



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.