Abaqus 6.11

Analysis User's Manual

Volume V: Prescribed Conditions,
Constraints & Interactions



Abaqus Analysis

User's Manual

Volume V

Legal Notices

CAUTION: This documentation is intended for qualified users who will exercise sound engineering judgment and expertise in the use of the Abaqus Software. The Abaqus Software is inherently complex, and the examples and procedures in this documentation are not intended to be exhaustive or to apply to any particular situation. Users are cautioned to satisfy themselves as to the accuracy and results of their analyses.

Dassault Systèmes and its subsidiaries, including Dassault Systèmes Simulia Corp., shall not be responsible for the accuracy or usefulness of any analysis performed using the Abaqus Software or the procedures, examples, or explanations in this documentation. Dassault Systèmes and its subsidiaries shall not be responsible for the consequences of any errors or omissions that may appear in this documentation.

The Abaqus Software is available only under license from Dassault Systèmes or its subsidiary and may be used or reproduced only in accordance with the terms of such license. This documentation is subject to the terms and conditions of either the software license agreement signed by the parties, or, absent such an 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.

No part of this documentation may be reproduced or distributed in any form without prior written permission of Dassault Systèmes or its subsidiary.

The Abaqus Software is a product of Dassault Systèmes Simulia Corp., Providence, RI, USA.

© Dassault Systèmes, 2011

Abaqus, the 3DS logo, SIMULIA, CATIA, and Unified FEA are trademarks or registered trademarks of Dassault Systèmes or its subsidiaries in the United States and/or other countries.

Other company, product, and service names may be trademarks or service marks of their respective owners. For additional information concerning trademarks, copyrights, and licenses, see the Legal Notices in the Abaqus 6.11 Release Notes.

Locations

SIMULIA Worldwide Headquarters	Rising Sun Mills, 166 Valley Street, Providence, RI 02909–2499, Tel: +1 401 276 4400, Fax: +1 401 276 4408, simulia.support@3ds.com, http://www.simulia.com
SIMULIA European Headquarters	Stationsplein 8-K, 6221 BT Maastricht, The Netherlands, Tel: +31 43 7999 084, Fax: +31 43 7999 306, simulia.europe.info@3ds.com

Dassault Systèmes' Centers of Simulation Excellence

United States	Fremont, CA, Tel: +1 510 794 5891, simulia.west.support@3ds.com
	West Lafayette, IN, Tel: +1 765 497 1373, simulia.central.support@3ds.com
	Northville, MI, Tel: +1 248 349 4669, simulia.greatlakes.info@3ds.com
	Woodbury, MN, Tel: +1 612 424 9044, simulia.central.support@3ds.com
	Beachwood, OH, Tel: +1 216 378 1070, simulia.erie.info@3ds.com
	West Chester, OH, Tel: +1 513 275 1430, simulia.central.support@3ds.com
	Warwick, RI, Tel: +1 401 739 3637, simulia.east.support@3ds.com
	Lewisville, TX, Tel: +1 972 221 6500, simulia.south.info@3ds.com
Australia	Richmond VIC, Tel: +61 3 9421 2900, simulia.au.support@3ds.com
Austria	Vienna, Tel: +43 1 22 707 200, simulia.at.info@3ds.com
Benelux	Maarssen, The Netherlands, Tel: +31 346 585 710, simulia.benelux.support@3ds.com
Canada	Toronto, ON, Tel: +1 416 402 2219, simulia.greatlakes.info@3ds.com
China	Beijing, P. R. China, Tel: +8610 6536 2288, simulia.cn.support@3ds.com
	Shanghai, P. R. China, Tel: +8621 3856 8000, simulia.cn.support@3ds.com
Finland	Espoo, Tel: +358 40 902 2973, simulia.nordic.info@3ds.com
France	Velizy Villacoublay Cedex, Tel: +33 1 61 62 72 72, simulia.fr.support@3ds.com
Germany	Aachen, Tel: +49 241 474 01 0, simulia.de.info@3ds.com
	Munich, Tel: +49 89 543 48 77 0, simulia.de.info@3ds.com
India	Chennai, Tamil Nadu, Tel: +91 44 43443000, simulia.in.info@3ds.com
Italy	Lainate MI, Tel: +39 02 334306150, simulia.ity.info@3ds.com
Japan	Tokyo, Tel: +81 3 5442 6300, simulia.tokyo.support@3ds.com
	Osaka, Tel: +81 6 7730 2703, simulia.osaka.support@3ds.com
	Yokohama-shi, Kanagawa, Tel: +81 45 470 9381, isight.jp.info@3ds.com
Korea	Mapo-Gu, Seoul, Tel: +82 2 785 6707/8, simulia.kr.info@3ds.com
Latin America	Puerto Madero, Buenos Aires, Tel: +54 11 4312 8700, Horacio.Burbridge@3ds.com
Scandinavia	Västerås, Sweden, Tel: +46 21 150870, simulia.nordic.info@3ds.com
United Kingdom	Warrington, Tel: +44 1925 830900, simulia.uk.info@3ds.com

Authorized Support Centers

Czech & Slovak Republics	Synerma s. r. o., Psáry, Prague-West, Tel: +420 603 145 769, abaqus@synerma.cz
Greece	3 Dimensional Data Systems, Crete, Tel: +30 2821040012, support@3dds.gr
Israel	ADCOM, Givataim, Tel: +972 3 7325311, shmulik.keidar@adcomsim.co.il
Malaysia	WorleyParsons Advanced Analysis, Kuala Lumpur, Tel: +603 2039 9000, abaqus.my@worleyparsons.com
New Zealand	Matrix Applied Computing Ltd., Auckland, Tel: +64 9 623 1223, abaqus-tech@matrix.co.nz
Poland	BudSoft Sp. z o.o., Poznań, Tel: +48 61 8508 466, info@budsoft.com.pl
Russia, Belarus & Ukraine	TESIS Ltd., Moscow, Tel: +7 495 612 44 22, info@tesis.com.ru
Singapore	WorleyParsons Advanced Analysis, Singapore, Tel: +65 6735 8444, abaqus.sg@worleyparsons.com
South Africa	Finite Element Analysis Services (Pty) Ltd., Parklands, Tel: +27 21 556 6462, feas@feas.co.za
Spain & Portugal	Principia Ingenieros Consultores, S.A., Madrid, Tel: +34 91 209 1482, simulia@principia.es
Taiwan	Simutech Solution Corporation, Taipei, R.O.C., Tel: +886 2 2507 9550, lucille@simutech.com.tw
Thailand	WorleyParsons Advanced Analysis, Singapore, Tel: +65 6735 8444, abaqus.sg@worleyparsons.com
Turkey	A-Ztech Ltd., Istanbul, Tel: +90 216 361 8850, info@a-ztech.com.tr

Complete contact information is available at http://www.simulia.com/locations/locations.html.

Preface

This section lists various resources that are available for help with using Abaqus Unified FEA software.

Support

Both technical engineering support (for problems with creating a model or performing an analysis) and systems support (for installation, licensing, and hardware-related problems) for Abaqus are offered through a network of local support offices. Regional contact information is listed in the front of each Abaqus manual and is accessible from the **Locations** page at www.simulia.com.

Support for SIMULIA products

SIMULIA provides a knowledge database of answers and solutions to questions that we have answered, as well as guidelines on how to use Abaqus, SIMULIA Scenario Definition, Isight, and other SIMULIA products. You can also submit new requests for support. All support incidents are tracked. If you contact us by means outside the system to discuss an existing support problem and you know the incident or support request number, please mention it so that we can consult the database to see what the latest action has been.

Many questions about Abaqus can also be answered by visiting the **Products** page and the **Support** page at www.simulia.com.

Anonymous ftp site

To facilitate data transfer with SIMULIA, an anonymous ftp account is available on the computer ftp.simulia.com. Login as user anonymous, and type your e-mail address as your password. Contact support before placing files on the site.

Training

All offices and representatives offer regularly scheduled public training classes. The courses are offered in a traditional classroom form and via the Web. We also provide training seminars at customer sites. All training classes and seminars include workshops to provide as much practical experience with Abaqus as possible. For a schedule and descriptions of available classes, see www.simulia.com or call your local office or representative.

Feedback

We welcome any suggestions for improvements to Abaqus software, the support program, or documentation. We will ensure that any enhancement requests you make are considered for future releases. If you wish to make a suggestion about the service or products, refer to www.simulia.com. Complaints should be addressed by contacting your local office or through www.simulia.com by visiting the **Quality Assurance** section of the **Support** page.

Contents

Volume I

PART I INTRODUCTION, SPATIAL MODELING, AND EXECUTION

1. Introduction

Introduction: general 1.1.1

Abaqus syntax and conventions

Input syntax rules 1.2.1
Conventions 1.2.2

Abaqus model definition

Defining a model in Abaqus 1.3.1

Parametric modeling

Parametric input 1.4.1

2. Spatial Modeling

Node definition

Node definition 2.1.1
Parametric shape variation 2.1.2
Nodal thicknesses 2.1.3
Normal definitions at nodes 2.1.4
Transformed coordinate systems 2.1.5
Adjust nodal coordinate 2.1.6

Element definition

Element definition 2.2.1
Element foundations 2.2.2
Defining reinforcement 2.2.3
Defining rebar as an element property 2.2.4
Orientations 2.2.5

Surface definition

Surfaces: overview 2.3.1
Element-based surface definition 2.3.2
Node-based surface definition 2.3.3
Analytical rigid surface definition 2.3.4

Eulerian surface definition 2.3.5
Operating on surfaces 2.3.6

Rigid body definition

Rigid body definition 2.4.1

Integrated output section definition

Integrated output section definition 2.5.1

Mass adjustment

Adjust and/or redistribute mass of an element set 2.6.1

Nonstructural mass definition

Nonstructural mass definition 2.7.1

Distribution definition

Distribution definition 2.8.1

Display body definition

Display body definition 2.9.1

Assembly definition

Defining an assembly 2.10.1

Matrix definition

Defining matrices 2.11.1

3. Job Execution

Execution procedures: overview

Execution procedure for Abaqus: overview 3.1.1

Execution procedures

Obtaining information 3.2.1
Abaqus/Standard, Abaqus/Explicit, and Abaqus/CFD execution 3.2.2
Abaqus/Standard, Abaqus/Explicit, and Abaqus/CFD co-simulation execution 3.2.3
Abaqus/CAE execution 3.2.4
Abaqus/Viewer execution 3.2.5
Python execution 3.2.6
Parametric studies 3.2.7
Abaqus documentation 3.2.8
Licensing utilities 3.2.9
ASCII translation of results (`.fil`) files 3.2.10
Joining results (`.fil`) files 3.2.11
Querying the keyword/problem database 3.2.12
Fetching sample input files 3.2.13

Making user-defined executables and subroutines 3.2.14
Input file and output database upgrade utility 3.2.15
Generating output database reports 3.2.16
Joining output database (**.odb**) files from restarted analyses 3.2.17
Combining output from substructures 3.2.18
Combining data from multiple output databases 3.2.19
Network output database file connector 3.2.20
Mapping thermal and magnetic loads 3.2.21
Fixed format conversion utility 3.2.22
Translating Nastran bulk data files to Abaqus input files 3.2.23
Translating Abaqus files to Nastran bulk data files 3.2.24
Translating ANSYS input files to Abaqus input files 3.2.25
Translating PAM-CRASH input files to partial Abaqus input files 3.2.26
Translating RADIOSS input files to partial Abaqus input files 3.2.27
Translating Abaqus output database files to Nastran Output2 results files 3.2.28
Exchanging Abaqus data with ZAERO 3.2.29
Encrypting and decrypting Abaqus input data 3.2.30
Job execution control 3.2.31

Environment file settings

Using the Abaqus environment settings 3.3.1

Managing memory and disk resources

Managing memory and disk use in Abaqus 3.4.1

Parallel execution

Parallel execution: overview 3.5.1
Parallel execution in Abaqus/Standard 3.5.2
Parallel execution in Abaqus/Explicit 3.5.3
Parallel execution in Abaqus/CFD 3.5.4

File extension definitions

File extensions used by Abaqus 3.6.1

FORTRAN unit numbers

FORTRAN unit numbers used by Abaqus 3.7.1

PART II OUTPUT

4. Output

Output 4.1.1
Output to the data and results files 4.1.2

Output to the output database 4.1.3
Error indicator output 4.1.4

Output variables

Abaqus/Standard output variable identifiers 4.2.1
Abaqus/Explicit output variable identifiers 4.2.2
Abaqus/CFD output variable identifiers 4.2.3

The postprocessing calculator

The postprocessing calculator 4.3.1

5. File Output Format

Accessing the results file

Accessing the results file: overview 5.1.1
Results file output format 5.1.2
Accessing the results file information 5.1.3
Utility routines for accessing the results file 5.1.4

OI.1 Abaqus/Standard Output Variable Index

OI.2 Abaqus/Explicit Output Variable Index

OI.3 Abaqus/CFD Output Variable Index

Volume II

PART III ANALYSIS PROCEDURES, SOLUTION, AND CONTROL

6. Analysis Procedures

Introduction

Procedures: overview 6.1.1
General and linear perturbation procedures 6.1.2
Multiple load case analysis 6.1.3
Direct linear equation solver 6.1.4
Iterative linear equation solver 6.1.5

Static stress/displacement analysis

Static stress analysis procedures: overview 6.2.1
Static stress analysis 6.2.2
Eigenvalue buckling prediction 6.2.3
Unstable collapse and postbuckling analysis 6.2.4
Quasi-static analysis 6.2.5
Direct cyclic analysis 6.2.6
Low-cycle fatigue analysis using the direct cyclic approach 6.2.7

Dynamic stress/displacement analysis

Dynamic analysis procedures: overview 6.3.1
Implicit dynamic analysis using direct integration 6.3.2
Explicit dynamic analysis 6.3.3
Direct-solution steady-state dynamic analysis 6.3.4
Natural frequency extraction 6.3.5
Complex eigenvalue extraction 6.3.6
Transient modal dynamic analysis 6.3.7
Mode-based steady-state dynamic analysis 6.3.8
Subspace-based steady-state dynamic analysis 6.3.9
Response spectrum analysis 6.3.10
Random response analysis 6.3.11

Steady-state transport analysis

Steady-state transport analysis 6.4.1

Heat transfer and thermal-stress analysis

Heat transfer analysis procedures: overview 6.5.1
Uncoupled heat transfer analysis 6.5.2
Sequentially coupled thermal-stress analysis 6.5.3

Fully coupled thermal-stress analysis 6.5.4
Adiabatic analysis 6.5.5

Fluid dynamic analysis

Fluid dynamic analysis procedures: overview 6.6.1
Incompressible fluid dynamic analysis 6.6.2

Electromagnetic analysis

Electromagnetic analysis procedures 6.7.1
Piezoelectric analysis 6.7.2
Coupled thermal-electrical analysis 6.7.3
Fully coupled thermal-electrical-structural analysis 6.7.4
Time-harmonic eddy current analysis 6.7.5

Coupled pore fluid flow and stress analysis

Coupled pore fluid diffusion and stress analysis 6.8.1
Geostatic stress state 6.8.2

Mass diffusion analysis

Mass diffusion analysis 6.9.1

Acoustic and shock analysis

Acoustic, shock, and coupled acoustic-structural analysis 6.10.1

Abaqus/Aqua analysis

Abaqus/Aqua analysis 6.11.1

Annealing

Annealing procedure 6.12.1

7. Analysis Solution and Control

Solving nonlinear problems

Solving nonlinear problems 7.1.1

Analysis convergence controls

Convergence and time integration criteria: overview 7.2.1
Commonly used control parameters 7.2.2
Convergence criteria for nonlinear problems 7.2.3
Time integration accuracy in transient problems 7.2.4

PART IV ANALYSIS TECHNIQUES

8. Analysis Techniques: Introduction

Analysis techniques: overview 8.1.1

9. Analysis Continuation Techniques

Restarting an analysis

Restarting an analysis 9.1.1

Importing and transferring results

Transferring results between Abaqus analyses: overview 9.2.1
Transferring results between Abaqus/Explicit and Abaqus/Standard 9.2.2
Transferring results from one Abaqus/Standard analysis to another 9.2.3
Transferring results from one Abaqus/Explicit analysis to another 9.2.4

10. Modeling Abstractions

Substructuring

Using substructures 10.1.1
Defining substructures 10.1.2

Submodeling

Submodeling: overview 10.2.1
Node-based submodeling 10.2.2
Surface-based submodeling 10.2.3

Generating global matrices

Generating matrices 10.3.1

Symmetric model generation, results transfer, and analysis of cyclic symmetry models

Symmetric model generation 10.4.1
Transferring results from a symmetric mesh or a partial three-dimensional mesh to a full three-dimensional mesh 10.4.2
Analysis of models that exhibit cyclic symmetry 10.4.3

Periodic media analysis

Periodic media analysis 10.5.1

Meshed beam cross-sections

Meshed beam cross-sections 10.6.1

Modeling discontinuities as an enriched feature using the extended finite element method

Modeling discontinuities as an enriched feature using the extended finite element method 10.7.1

11. Special-Purpose Techniques

Inertia relief

Inertia relief 11.1.1

Mesh modification or replacement

Element and contact pair removal and reactivation 11.2.1

Geometric imperfections

Introducing a geometric imperfection into a model 11.3.1

Fracture mechanics

Fracture mechanics: overview 11.4.1
Contour integral evaluation 11.4.2
Crack propagation analysis 11.4.3

Surface-based fluid modeling

Surface-based fluid cavities: overview 11.5.1
Fluid cavity definition 11.5.2
Fluid exchange definition 11.5.3
Inflator definition 11.5.4

Mass scaling

Mass scaling 11.6.1

Selective subcycling

Selective subcycling 11.7.1

Steady-state detection

Steady-state detection 11.8.1

12. Adaptivity Techniques

Adaptivity techniques: overview

Adaptivity techniques 12.1.1

ALE adaptive meshing

ALE adaptive meshing: overview 12.2.1
Defining ALE adaptive mesh domains in Abaqus/Explicit 12.2.2
ALE adaptive meshing and remapping in Abaqus/Explicit 12.2.3
Modeling techniques for Eulerian adaptive mesh domains in Abaqus/Explicit 12.2.4

Output and diagnostics for ALE adaptive meshing in Abaqus/Explicit 12.2.5
Defining ALE adaptive mesh domains in Abaqus/Standard 12.2.6
ALE adaptive meshing and remapping in Abaqus/Standard 12.2.7

Adaptive remeshing

Adaptive remeshing: overview 12.3.1
Selection of error indicators influencing adaptive remeshing 12.3.2
Solution-based mesh sizing 12.3.3

Analysis continuation after mesh replacement

Mesh-to-mesh solution mapping 12.4.1

13. Optimization Techniques

Structural optimization: overview

Structural optimization: overview 13.1.1

Optimization models

Design responses 13.2.1
Objectives and constraints 13.2.2
Creating Abaqus optimization models 13.2.3

14. Eulerian Analysis

Eulerian analysis 14.1.1
Defining Eulerian boundaries 14.1.2
Eulerian mesh motion 14.1.3

15. Particle Methods

Smoothed particle hydrodynamic analyses

Smoothed particle hydrodynamic analysis 15.1.1

16. Multiphysics Analyses

Co-simulation

Co-simulation: overview 16.1.1
Preparing an Abaqus analysis for co-simulation 16.1.2
Rendezvousing schemes for coupling Abaqus to third-party analysis programs 16.1.3
Abaqus/Standard to Abaqus/Explicit co-simulation 16.1.4
Abaqus/CFD to Abaqus/Standard or to Abaqus/Explicit co-simulation 16.1.5

Sequentially coupled multiphysics analyses

Sequentially coupled multiphysics analyses using predefined fields 16.2.1
Sequentially coupled multiphysics analyses using predefined loads 16.2.2

17. Extending Abaqus Analysis Functionality

User subroutines and utilities

User subroutines: overview 17.1.1
Available user subroutines 17.1.2
Available utility routines 17.1.3

18. Design Sensitivity Analysis

Design sensitivity analysis 18.1.1

19. Parametric Studies

Scripting parametric studies

Scripting parametric studies 19.1.1

Parametric studies: commands

aStudy.combine(): Combine parameter samples for parametric studies. 19.2.1
aStudy.constrain(): Constrain parameter value combinations in parametric studies. 19.2.2
aStudy.define(): Define parameters for parametric studies. 19.2.3
aStudy.execute(): Execute the analysis of parametric study designs. 19.2.4
aStudy.gather(): Gather the results of a parametric study. 19.2.5
aStudy.generate(): Generate the analysis job data for a parametric study. 19.2.6
aStudy.output(): Specify the source of parametric study results. 19.2.7
aStudy=ParStudy(): Create a parametric study. 19.2.8
aStudy.report(): Report parametric study results. 19.2.9
aStudy.sample(): Sample parameters for parametric studies. 19.2.10

Volume III

PART V MATERIALS

20. Materials: Introduction

Introduction

Material library: overview 20.1.1
Material data definition 20.1.2
Combining material behaviors 20.1.3

General properties

Density 20.2.1

21. Elastic Mechanical Properties

Overview

Elastic behavior: overview 21.1.1

Linear elasticity

Linear elastic behavior 21.2.1
No compression or no tension 21.2.2
Plane stress orthotropic failure measures 21.2.3

Porous elasticity

Elastic behavior of porous materials 21.3.1

Hypoelasticity

Hypoelastic behavior 21.4.1

Hyperelasticity

Hyperelastic behavior of rubberlike materials 21.5.1
Hyperelastic behavior in elastomeric foams 21.5.2
Anisotropic hyperelastic behavior 21.5.3

Stress softening in elastomers

Mullins effect 21.6.1
Energy dissipation in elastomeric foams 21.6.2

Viscoelasticity

Time domain viscoelasticity 21.7.1
Frequency domain viscoelasticity 21.7.2

Hysteresis

Hysteresis in elastomers 21.8.1

Rate sensitive elastomeric foams

Low-density foams 21.9.1

22. Inelastic Mechanical Properties

Overview

Inelastic behavior 22.1.1

Metal plasticity

Classical metal plasticity 22.2.1
Models for metals subjected to cyclic loading 22.2.2
Rate-dependent yield 22.2.3
Rate-dependent plasticity: creep and swelling 22.2.4
Annealing or melting 22.2.5
Anisotropic yield/creep 22.2.6
Johnson-Cook plasticity 22.2.7
Dynamic failure models 22.2.8
Porous metal plasticity 22.2.9
Cast iron plasticity 22.2.10
Two-layer viscoplasticity 22.2.11
ORNL – Oak Ridge National Laboratory constitutive model 22.2.12
Deformation plasticity 22.2.13

Other plasticity models

Extended Drucker-Prager models 22.3.1
Modified Drucker-Prager/Cap model 22.3.2
Mohr-Coulomb plasticity 22.3.3
Critical state (clay) plasticity model 22.3.4
Crushable foam plasticity models 22.3.5

Fabric materials

Fabric material behavior 22.4.1

Jointed materials

Jointed material model 22.5.1

Concrete

Concrete smeared cracking 22.6.1
Cracking model for concrete 22.6.2
Concrete damaged plasticity 22.6.3

Permanent set in rubberlike materials

Permanent set in rubberlike materials 22.7.1

23. Progressive Damage and Failure

Progressive damage and failure: overview

Progressive damage and failure 23.1.1

Damage and failure for ductile metals

Damage and failure for ductile metals: overview 23.2.1
Damage initiation for ductile metals 23.2.2
Damage evolution and element removal for ductile metals 23.2.3

Damage and failure for fiber-reinforced composites

Damage and failure for fiber-reinforced composites: overview 23.3.1
Damage initiation for fiber-reinforced composites 23.3.2
Damage evolution and element removal for fiber-reinforced composites 23.3.3

Damage and failure for ductile materials in low-cycle fatigue analysis

Damage and failure for ductile materials in low-cycle fatigue analysis: overview 23.4.1
Damage initiation for ductile materials in low-cycle fatigue 23.4.2
Damage evolution for ductile materials in low-cycle fatigue 23.4.3

24. Hydrodynamic Properties

Overview

Hydrodynamic behavior: overview 24.1.1

Equations of state

Equation of state 24.2.1

25. Other Material Properties

Mechanical properties

Material damping 25.1.1
Thermal expansion 25.1.2
Field expansion 25.1.3
Viscosity 25.1.4

Heat transfer properties

Thermal properties: overview 25.2.1
Conductivity 25.2.2
Specific heat 25.2.3
Latent heat 25.2.4

Acoustic properties

Acoustic medium 25.3.1

Mass diffusion properties

Diffusivity 25.4.1
Solubility 25.4.2

Electromagnetic properties

Electrical conductivity 25.5.1
Piezoelectric behavior 25.5.2
Magnetic permeability 25.5.3

Pore fluid flow properties

Pore fluid flow properties 25.6.1
Permeability 25.6.2
Porous bulk moduli 25.6.3
Sorption 25.6.4
Swelling gel 25.6.5
Moisture swelling 25.6.6

User materials

User-defined mechanical material behavior 25.7.1
User-defined thermal material behavior 25.7.2

Volume IV

PART VI ELEMENTS

26. Elements: Introduction

Element library: overview 26.1.1
Choosing the element's dimensionality 26.1.2
Choosing the appropriate element for an analysis type 26.1.3
Section controls 26.1.4

27. Continuum Elements

General-purpose continuum elements

Solid (continuum) elements 27.1.1
One-dimensional solid (link) element library 27.1.2
Two-dimensional solid element library 27.1.3
Three-dimensional solid element library 27.1.4
Cylindrical solid element library 27.1.5
Axisymmetric solid element library 27.1.6
Axisymmetric solid elements with nonlinear, asymmetric deformation 27.1.7

Fluid continuum elements

Fluid (continuum) elements 27.2.1
Fluid element library 27.2.2

Infinite elements

Infinite elements 27.3.1
Infinite element library 27.3.2

Warping elements

Warping elements 27.4.1
Warping element library 27.4.2

Particle elements

Particle elements 27.5.1
Particle element library 27.5.2

28. Structural Elements

Membrane elements

Membrane elements 28.1.1
General membrane element library 28.1.2

Cylindrical membrane element library 28.1.3
Axisymmetric membrane element library 28.1.4

Truss elements

Truss elements 28.2.1
Truss element library 28.2.2

Beam elements

Beam modeling: overview 28.3.1
Choosing a beam cross-section 28.3.2
Choosing a beam element 28.3.3
Beam element cross-section orientation 28.3.4
Beam section behavior 28.3.5
Using a beam section integrated during the analysis to define the section behavior 28.3.6
Using a general beam section to define the section behavior 28.3.7
Beam element library 28.3.8
Beam cross-section library 28.3.9

Frame elements

Frame elements 28.4.1
Frame section behavior 28.4.2
Frame element library 28.4.3

Elbow elements

Pipes and pipebends with deforming cross-sections: elbow elements 28.5.1
Elbow element library 28.5.2

Shell elements

Shell elements: overview 28.6.1
Choosing a shell element 28.6.2
Defining the initial geometry of conventional shell elements 28.6.3
Shell section behavior 28.6.4
Using a shell section integrated during the analysis to define the section behavior 28.6.5
Using a general shell section to define the section behavior 28.6.6
Three-dimensional conventional shell element library 28.6.7
Continuum shell element library 28.6.8
Axisymmetric shell element library 28.6.9
Axisymmetric shell elements with nonlinear, asymmetric deformation 28.6.10

29. Inertial, Rigid, and Capacitance Elements

Point mass elements

Point masses 29.1.1
Mass element library 29.1.2

Rotary inertia elements

Rotary inertia 29.2.1
Rotary inertia element library 29.2.2

Rigid elements

Rigid elements 29.3.1
Rigid element library 29.3.2

Capacitance elements

Point capacitance 29.4.1
Capacitance element library 29.4.2

30. Connector Elements

Connector elements

Connectors: overview 30.1.1
Connector elements 30.1.2
Connector actuation 30.1.3
Connector element library 30.1.4
Connection-type library 30.1.5

Connector element behavior

Connector behavior 30.2.1
Connector elastic behavior 30.2.2
Connector damping behavior 30.2.3
Connector functions for coupled behavior 30.2.4
Connector friction behavior 30.2.5
Connector plastic behavior 30.2.6
Connector damage behavior 30.2.7
Connector stops and locks 30.2.8
Connector failure behavior 30.2.9
Connector uniaxial behavior 30.2.10

31. Special-Purpose Elements

Spring elements

Springs 31.1.1
Spring element library 31.1.2

Dashpot elements

Dashpots 31.2.1
Dashpot element library 31.2.2

Flexible joint elements

Flexible joint element 31.3.1
Flexible joint element library 31.3.2

Distributing coupling elements

Distributing coupling elements 31.4.1
Distributing coupling element library 31.4.2

Cohesive elements

Cohesive elements: overview 31.5.1
Choosing a cohesive element 31.5.2
Modeling with cohesive elements 31.5.3
Defining the cohesive element's initial geometry 31.5.4
Defining the constitutive response of cohesive elements using a continuum approach 31.5.5
Defining the constitutive response of cohesive elements using a traction-separation description 31.5.6
Defining the constitutive response of fluid within the cohesive element gap 31.5.7
Two-dimensional cohesive element library 31.5.8
Three-dimensional cohesive element library 31.5.9
Axisymmetric cohesive element library 31.5.10

Gasket elements

Gasket elements: overview 31.6.1
Choosing a gasket element 31.6.2
Including gasket elements in a model 31.6.3
Defining the gasket element's initial geometry 31.6.4
Defining the gasket behavior using a material model 31.6.5
Defining the gasket behavior directly using a gasket behavior model 31.6.6
Two-dimensional gasket element library 31.6.7
Three-dimensional gasket element library 31.6.8
Axisymmetric gasket element library 31.6.9

Surface elements

Surface elements 31.7.1
General surface element library 31.7.2
Cylindrical surface element library 31.7.3
Axisymmetric surface element library 31.7.4

Tube support elements

Tube support elements 31.8.1
Tube support element library 31.8.2

Line spring elements

Line spring elements for modeling part-through cracks in shells 31.9.1
Line spring element library 31.9.2

Elastic-plastic joints

Elastic-plastic joints 31.10.1
Elastic-plastic joint element library 31.10.2

Drag chain elements

Drag chains 31.11.1
Drag chain element library 31.11.2

Pipe-soil elements

Pipe-soil interaction elements 31.12.1
Pipe-soil interaction element library 31.12.2

Acoustic interface elements

Acoustic interface elements 31.13.1
Acoustic interface element library 31.13.2

Eulerian elements

Eulerian elements 31.14.1
Eulerian element library 31.14.2

User-defined elements

User-defined elements 31.15.1
User-defined element library 31.15.2

EI.1 Abaqus/Standard Element Index

EI.2 Abaqus/Explicit Element Index

EI.3 Abaqus/CFD Element Index

Volume V

PART VII PRESCRIBED CONDITIONS

32. Prescribed Conditions

Overview

Prescribed conditions: overview 32.1.1
Amplitude curves 32.1.2

Initial conditions

Initial conditions in Abaqus/Standard and Abaqus/Explicit 32.2.1
Initial conditions in Abaqus/CFD 32.2.2

Boundary conditions

Boundary conditions in Abaqus/Standard and Abaqus/Explicit 32.3.1
Boundary conditions in Abaqus/CFD 32.3.2

Loads

Applying loads: overview 32.4.1
Concentrated loads 32.4.2
Distributed loads 32.4.3
Thermal loads 32.4.4
Electromagnetic loads 32.4.5
Acoustic and shock loads 32.4.6
Pore fluid flow 32.4.7

Prescribed assembly loads

Prescribed assembly loads 32.5.1

Predefined fields

Predefined fields 32.6.1

PART VIII CONSTRAINTS

33. Constraints

Overview

Kinematic constraints: overview 33.1.1

Multi-point constraints

Linear constraint equations 33.2.1

General multi-point constraints 33.2.2
Kinematic coupling constraints 33.2.3

Surface-based constraints

Mesh tie constraints 33.3.1
Coupling constraints 33.3.2
Shell-to-solid coupling 33.3.3
Mesh-independent fasteners 33.3.4

Embedded elements

Embedded elements 33.4.1

Element end release

Element end release 33.5.1

Overconstraint checks

Overconstraint checks 33.6.1

PART IX INTERACTIONS

34. Defining Contact Interactions

Overview

Contact interaction analysis: overview 34.1.1

Defining general contact in Abaqus/Standard

Defining general contact interactions in Abaqus/Standard 34.2.1
Surface properties for general contact in Abaqus/Standard 34.2.2
Contact properties for general contact in Abaqus/Standard 34.2.3
Controlling initial contact status in Abaqus/Standard 34.2.4
Stabilization for general contact in Abaqus/Standard 34.2.5
Numerical controls for general contact in Abaqus/Standard 34.2.6

Defining contact pairs in Abaqus/Standard

Defining contact pairs in Abaqus/Standard 34.3.1
Assigning surface properties for contact pairs in Abaqus/Standard 34.3.2
Assigning contact properties for contact pairs in Abaqus/Standard 34.3.3
Modeling contact interference fits in Abaqus/Standard 34.3.4
Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs 34.3.5
Adjusting contact controls in Abaqus/Standard 34.3.6
Defining tied contact in Abaqus/Standard 34.3.7
Extending master surfaces and slide lines 34.3.8

Contact modeling if substructures are present 34.3.9
Contact modeling if asymmetric-axisymmetric elements are present 34.3.10

Defining general contact in Abaqus/Explicit

Defining general contact interactions in Abaqus/Explicit 34.4.1
Assigning surface properties for general contact in Abaqus/Explicit 34.4.2
Assigning contact properties for general contact in Abaqus/Explicit 34.4.3
Controlling initial contact status for general contact in Abaqus/Explicit 34.4.4
Contact controls for general contact in Abaqus/Explicit 34.4.5

Defining contact pairs in Abaqus/Explicit

Defining contact pairs in Abaqus/Explicit 34.5.1
Assigning surface properties for contact pairs in Abaqus/Explicit 34.5.2
Assigning contact properties for contact pairs in Abaqus/Explicit 34.5.3
Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit 34.5.4
Contact controls for contact pairs in Abaqus/Explicit 34.5.5

35. Contact Property Models

Mechanical contact properties

Mechanical contact properties: overview 35.1.1
Contact pressure-overclosure relationships 35.1.2
Contact damping 35.1.3
Contact blockage 35.1.4
Frictional behavior 35.1.5
User-defined interfacial constitutive behavior 35.1.6
Pressure penetration loading 35.1.7
Interaction of debonded surfaces 35.1.8
Breakable bonds 35.1.9
Surface-based cohesive behavior 35.1.10

Thermal contact properties

Thermal contact properties 35.2.1

Electrical contact properties

Electrical contact properties 35.3.1

Pore fluid contact properties

Pore fluid contact properties 35.4.1

36. Contact Formulations and Numerical Methods

Contact formulations and numerical methods in Abaqus/Standard

Contact formulations in Abaqus/Standard 36.1.1

Contact constraint enforcement methods in Abaqus/Standard 36.1.2
Smoothing contact surfaces in Abaqus/Standard 36.1.3

Contact formulations and numerical methods in Abaqus/Explicit

Contact formulation for general contact in Abaqus/Explicit 36.2.1
Contact formulations for contact pairs in Abaqus/Explicit 36.2.2
Contact constraint enforcement methods in Abaqus/Explicit 36.2.3

37. Contact Difficulties and Diagnostics

Resolving contact difficulties in Abaqus/Standard

Contact diagnostics in an Abaqus/Standard analysis 37.1.1
Common difficulties associated with contact modeling in Abaqus/Standard 37.1.2

Resolving contact difficulties in Abaqus/Explicit

Contact diagnositcs in an Abaqus/Explicit analysis 37.2.1
Common difficulties associated with contact modeling using contact pairs in Abaqus/Explicit 37.2.2

38. Contact Elements in Abaqus/Standard

Contact modeling with elements

Contact modeling with elements 38.1.1

Gap contact elements

Gap contact elements 38.2.1
Gap element library 38.2.2

Tube-to-tube contact elements

Tube-to-tube contact elements 38.3.1
Tube-to-tube contact element library 38.3.2

Slide line contact elements

Slide line contact elements 38.4.1
Axisymmetric slide line element library 38.4.2

Rigid surface contact elements

Rigid surface contact elements 38.5.1
Axisymmetric rigid surface contact element library 38.5.2

39. Defining Cavity Radiation in Abaqus/Standard

Cavity radiation 39.1.1

Part VII: Prescribed Conditions

- Chapter 32, “Prescribed Conditions”

32. Prescribed Conditions

Overview 32.1
Initial conditions 32.2
Boundary conditions 32.3
Loads 32.4
Prescribed assembly loads 32.5
Predefined fields 32.6

32.1 Overview

- “Prescribed conditions: overview,” Section 32.1.1
- “Amplitude curves,” Section 32.1.2

32.1.1 PRESCRIBED CONDITIONS: OVERVIEW

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

- **Initial conditions:** Nonzero initial conditions can be defined for many variables, as described in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1, and "Initial conditions in Abaqus/CFD," Section 32.2.2.
- **Boundary conditions:** Boundary conditions are used to prescribe values of basic solution variables: displacements and rotations in stress/displacement analysis, temperature in heat transfer or coupled thermal-stress analysis, electrical potential in coupled thermal-electrical analysis, pore pressure in soils analysis, acoustic pressure in acoustic analysis, etc. Boundary conditions can be defined as described in "Boundary conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.3.1, and "Boundary conditions in Abaqus/CFD," Section 32.3.2.
- **Loads:** Many types of loading are available, depending on the analysis procedure. "Applying loads: overview," Section 32.4.1, gives an overview of loading in Abaqus. Load types specific to one analysis procedure are described in the appropriate procedure section in Part III, "Analysis Procedures, Solution, and Control." General loads, which can be applied in multiple analysis types, are described in:
  - "Concentrated loads," Section 32.4.2
  - "Distributed loads," Section 32.4.3
  - "Thermal loads," Section 32.4.4
  - "Electromagnetic loads," Section 32.4.5
  - "Acoustic and shock loads," Section 32.4.6
  - "Pore fluid flow," Section 32.4.7
- **Prescribed assembly loads:** Pre-tension sections can be defined in Abaqus/Standard to prescribe assembly loads in bolts or any other type of fastener. Pre-tension sections are described in "Prescribed assembly loads," Section 32.5.1.
- **Connector loads and motions:** Connector elements can be used to define complex mechanical connections between parts, including actuation with prescribed loads or motions. Connector elements are described in "Connectors: overview," Section 30.1.1.
- **Predefined fields:** Predefined fields are time-dependent, non-solution-dependent fields that exist over the spatial domain of the model. Temperature is the most commonly defined field. Predefined fields are described in "Predefined fields," Section 32.6.1.

Amplitude variations

Complex time- or frequency-dependent boundary conditions, loads, and predefined fields can be specified by referring to an amplitude curve in the prescribed condition definition. Amplitude curves are explained in "Amplitude curves," Section 32.1.2.

In Abaqus/Standard if no amplitude is referenced from the boundary condition, loading, or predefined field definition, the total magnitude can be applied instantaneously at the start of the step and

remain constant throughout the step (a "step" variation) or it can vary linearly over the step from the value at the end of the previous step (or from zero at the start of the analysis) to the magnitude given (a "ramp" variation). You choose the type of variation when you define the step; the default variation depends on the procedure chosen, as shown in "Procedures: overview," Section 6.1.1.

In Abaqus/Standard the variation of many prescribed conditions can be defined in user subroutines. In this case the magnitude of the variable can vary in any way with position and time. The magnitude variation for prescribing and removing conditions must be specified in the subroutine (see "User subroutines and utilities," Section 17.1").

In Abaqus/Explicit if no amplitude is referenced from the boundary condition or loading definition, the total value will be applied instantaneously at the start of the step and will remain constant throughout the step (a "step" variation), although Abaqus/Explicit does not admit jumps in displacement (see "Boundary conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.3.1). If no amplitude is referenced from a predefined field definition, the total magnitude will vary linearly over the step from the value at the end of the previous step (or from zero at the start of the analysis) to the magnitude given (a "ramp" variation).

When boundary conditions are removed (see "Boundary conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.3.1), the boundary condition (displacement or rotation constraint in stress/displacement analysis) is converted to an applied conjugate flux (force or moment in stress/displacement analysis) at the beginning of the step. This flux magnitude is set to zero with a "step" or "ramp" variation depending on the procedure chosen, as discussed in "Procedures: overview," Section 6.1.1. Similarly, when loads and predefined fields are removed, the load is set to zero and the predefined field is set to its initial value.

In Abaqus/CFD if no amplitude is referenced from the boundary or loading condition, the total value is applied instantaneously at the start of the step and remains constant throughout the step. Abaqus/CFD does admit jumps in the velocity, temperature, etc. from the end value of the previous step to the magnitude given in the current step. However, jumps in velocity boundary conditions may result in a divergence-free projection that adjusts the initial velocities to be consistent with the prescribed boundary conditions in order to define a well-posed incompressible flow problem.

Applying boundary conditions and loads in a local coordinate system

You can define a local coordinate system at a node as described in "Transformed coordinate systems," Section 2.1.5. Then, all input data for concentrated force and moment loading and for displacement and rotation boundary conditions are given in the local system.

Loads and predefined fields available for various procedures

Table 32.1.1–1 Available loads and predefined fields.

Loads and predefined fields	Procedures
Added mass (concentrated and distributed)	Abaqus/Aqua eigenfrequency extraction analysis ("Natural frequency extraction," Section 6.3.5)
Base motion	Procedures based on eigenmodes: "Transient modal dynamic analysis," Section 6.3.7 "Mode-based steady-state dynamic analysis," Section 6.3.8 "Response spectrum analysis," Section 6.3.10 "Random response analysis," Section 6.3.11
Boundary condition with a nonzero prescribed boundary	All procedures except those based on eigenmodes
Connector motion Connector load	All relevant procedures except modal extraction, buckling, those based on eigenmodes, and direct steady-state dynamics
Cross-correlation property	"Random response analysis," Section 6.3.11
Current density (concentrated and distributed)	"Coupled thermal-electrical analysis," Section 6.7.3 "Fully coupled thermal-electrical-structural analysis," Section 6.7.4
Current density vector	"Time-harmonic eddy current analysis," Section 6.7.5
Electric charge (concentrated and distributed)	"Piezoelectric analysis," Section 6.7.2
Equivalent pressure stress	"Mass diffusion analysis," Section 6.9.1
Film coefficient and associated sink temperature	All procedures involving temperature degrees of freedom
Fluid flux	Analysis involving hydrostatic fluid elements
Fluid mass flow rate	Analysis involving convective heat transfer elements
Flux (concentrated and distributed)	All procedures involving temperature degrees of freedom "Mass diffusion analysis," Section 6.9.1
Force and moment (concentrated and distributed)	All procedures with displacement degrees of freedom except response spectrum

Loads and predefined fields	Procedures
Incident wave loading	Direct-integration dynamic analysis (“Implicit dynamic analysis using direct integration,” Section 6.3.2) involving solid and/or fluid elements undergoing shock loading
Predefined field variable	All procedures except those based on eigenmodes
Seepage coefficient and associated sink pore pressure Distributed seepage flow	“Coupled pore fluid diffusion and stress analysis,” Section 6.8.1
Substructure load	All procedures involving the use of substructures
Temperature as a predefined field	All procedures except adiabatic analysis, mode-based procedures, and procedures involving temperature degrees of freedom

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

32.1.2 AMPLITUDE CURVES

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CFD Abaqus/CAE

References

- "Prescribed conditions: overview," Section 32.1.1
- *AMPLITUDE
- Chapter 57, "The Amplitude toolset," of the Abaqus/CAE User's Manual

Overview

An amplitude curve:

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

Amplitude curves

By default, the values of loads, boundary conditions, and predefined fields either change linearly with time throughout the step (ramp function) or they are applied immediately and remain constant throughout the step (step function)—see "Procedures: overview," Section 6.1.1. Many problems require a more elaborate definition, however. For example, different amplitude curves can be used to specify time variations for different loadings. One common example is the combination of thermal and mechanical load transients: usually the temperatures and mechanical loads have different time variations during the step. Different amplitude curves can be used to specify each of these time variations.

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

Amplitudes are defined as model data (i.e., they are not step dependent). Each amplitude curve must be named; this name is then referred to from the load, boundary condition, or predefined field definition (see "Prescribed conditions: overview," Section 32.1.1).

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

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

Defining the time period

Each amplitude curve is a function of time or frequency. Amplitudes defined as functions of frequency are used in "Direct-solution steady-state dynamic analysis," Section 6.3.4; "Mode-based steady-state dynamic analysis," Section 6.3.8; and "Time-harmonic eddy current analysis," Section 6.7.5.

Amplitudes defined as functions of time can be given in terms of *step time* (default) or in terms of *total time*. These time measures are defined in "Conventions," Section 1.2.2.

Input File Usage: Use one of the following options:

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

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

Continuation of an amplitude reference in subsequent steps

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

- If the associated amplitude was given in terms of total time, the prescribed condition continues to follow the amplitude definition.
- If no associated amplitude was given or if the amplitude was given in terms of step time, the prescribed condition remains constant at the magnitude associated with the end of the previous step.

Specifying relative or absolute data

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

Relative data

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

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

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

Absolute data

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

Absolute amplitude values should generally not be used to define temperatures or predefined field variables for nodes attached to beam or shell elements as values at the reference surface together with the gradient or gradients across the section (default cross-section definition; see "Using a beam section integrated during the analysis to define the section behavior," Section 28.3.6, and "Using a shell section

integrated during the analysis to define the section behavior," Section 28.6.5). Because the values given in temperature fields and predefined fields are ignored, the absolute amplitude value will be used to define both the temperature and the gradient and field and gradient, respectively.

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

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

Defining the amplitude data

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

Defining tabular data

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

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

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

Several examples of tabular input are shown in Figure 32.1.2–1.

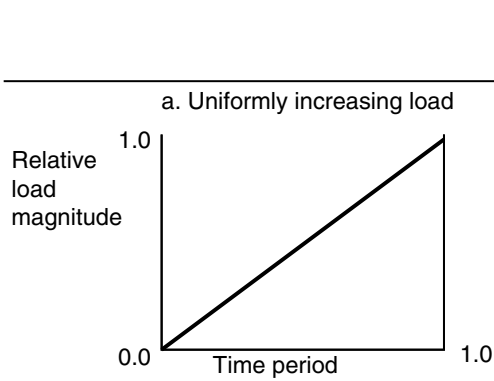
Input File Usage: *AMPLITUDE, NAME=*name*, DEFINITION=TABULAR

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

Defining equally spaced data

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

If the analysis time in a step is less than the earliest time for which data exist in the table, Abaqus applies the earliest value in the table for all step times less than the earliest tabulated time. Similarly,



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

Figure 32.1.2–1 Tabular amplitude definition examples.

if the analysis continues for step times past the last time for which data are defined in the table, the last value in the table is applied for all subsequent time.

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

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

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

Defining periodic data

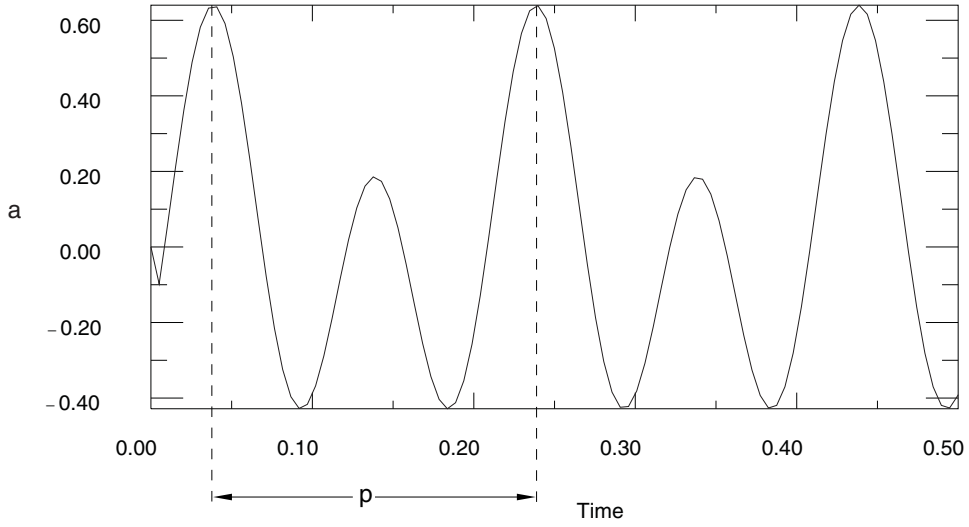
Choose the periodic definition method to define the amplitude, a, as a Fourier series:

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

where t_0, N, ω, A_0, A_n, and B_n, $n = 1, 2 \ldots N$, are user-defined constants. An example of this form of input is shown in Figure 32.1.2–2.

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

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



p = 0.2s

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

$a = A_0$ for $t < t_0$

with

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

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

Figure 32.1.2–2 Periodic amplitude definition example.

Defining modulated data

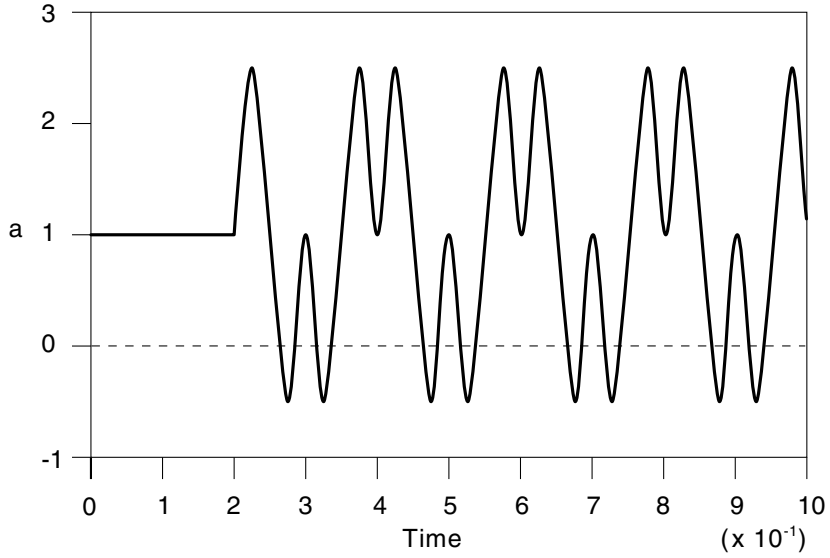
Choose the modulated definition method to define the amplitude, a, as

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

where A_0, A, t_0, ω_1, and ω_2 are user-defined constants. An example of this form of input is shown in Figure 32.1.2–3.

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

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



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

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

with

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

Figure 32.1.2–3 Modulated amplitude definition example.

Defining exponential decay

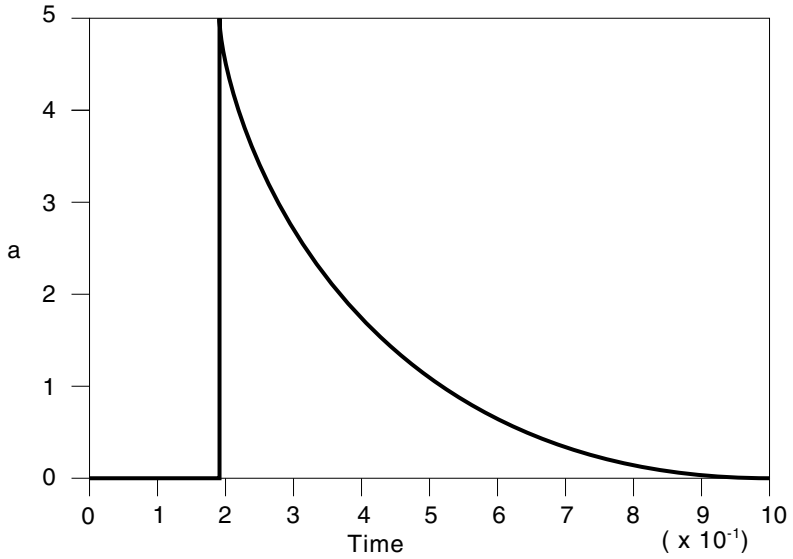
Choose the exponential decay definition method to define the amplitude, a, as

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

where A_0, A, t_0, and t_d are user-defined constants. An example of this form of input is shown in Figure 32.1.2–4.

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

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



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

$a = A_0$ for $t < t_0$

with

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

Figure 32.1.2–4 Exponential decay amplitude definition example.

Defining smooth step data

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

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

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

The amplitude, a, is defined such that

$$a = A_0 \qquad \text{for} \quad t \le t_0,$$
$$a = A_f \qquad \text{for} \quad t \ge t_f,$$

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

Examples of this form of input are shown in Figure 32.1.2–5 and Figure 32.1.2–6. This definition cannot be used to interpolate smoothly between a set of data points; i.e., this definition cannot be used to do curve fitting.

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

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

Defining a solution-dependent amplitude for superplastic forming analysis

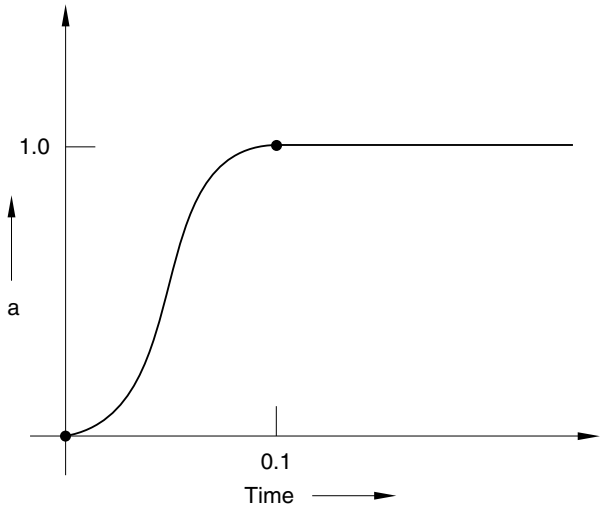
Abaqus/Standard can calculate amplitude values based on a solution-dependent variable. Choose the solution-dependent definition method to create a solution-dependent amplitude curve. The data consist of an initial value, a minimum value, and a maximum value. The amplitude starts with the initial value and is then modified based on the progress of the solution, subject to the minimum and maximum values. The maximum value is typically the controlling mechanism used to end the analysis. This method is used with creep strain rate control for superplastic forming analysis (see "Rate-dependent plasticity: creep and swelling," Section 22.2.4).

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

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

Defining the bubble load amplitude for an underwater explosion

Two interfaces are available in Abaqus for applying incident wave loads (see "Incident wave loading due to external sources" in "Acoustic and shock loads," Section 32.4.6). For either interface bubble dynamics can be described using a model internal to Abaqus. A description of this built-in mechanical model and the parameters that define the bubble behavior are discussed in "Defining bubble loading for spherical incident wave loading" in "Acoustic and shock loads," Section 32.4.6. The related theoretical details are described in "Loading due to an incident dilatational wave field," Section 6.3.1 of the Abaqus Theory Manual.



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

$a = A_0$ for $t \le t_0$

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

$= A_1$ for $t \ge t_1$

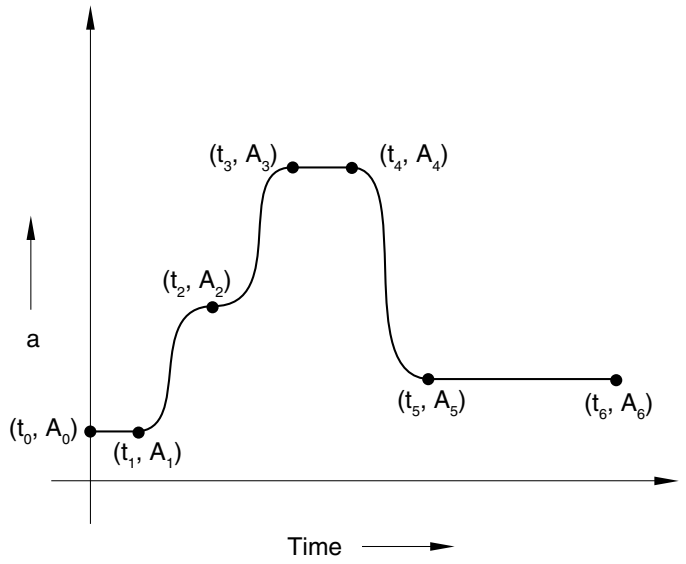
where $\xi = \dfrac{t - t_0}{t_1 - t_0}$

Figure 32.1.2–5 Smooth step amplitude definition example with two data points.

The preferred interface for incident wave loading due to an underwater explosion specifies bubble dynamics using the UNDEX charge property definition (see "Defining bubble loading for spherical incident wave loading" in "Acoustic and shock loads," Section 32.4.6). The alternative interface for incident wave loading uses the bubble definition described in this section to define bubble load amplitude curves.

An example of the bubble amplitude definition with the following input data is shown in Figure 32.1.2–7.

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



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

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

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

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

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

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

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

Figure 32.1.2–6 Smooth step amplitude definition example with multiple data points.

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

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

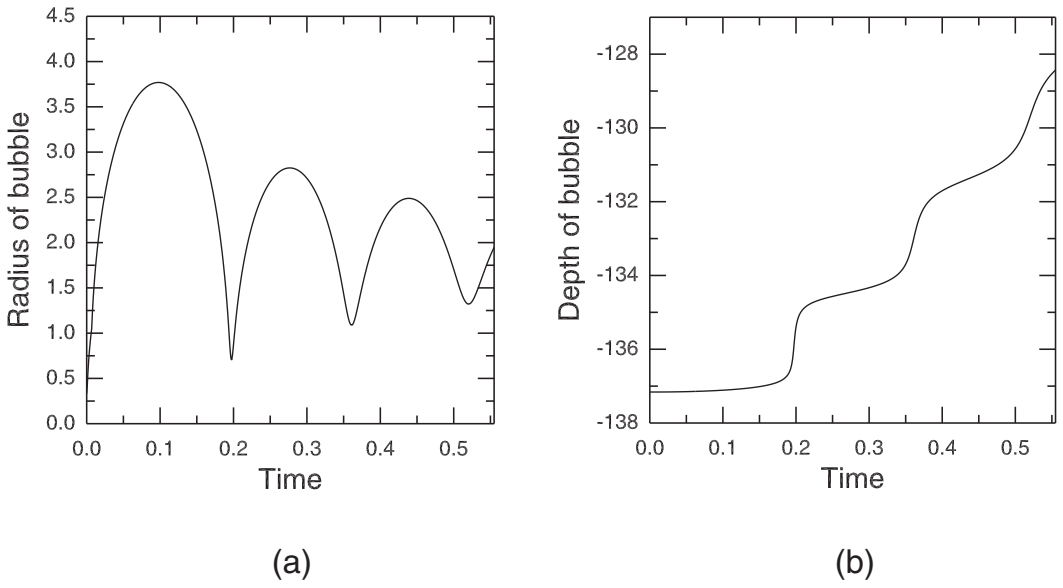


Figure 32.1.2–7 Bubble amplitude definition example: (a) radius of bubble and (b) depth of bubble center under fluid surface.

Defining an amplitude via a user subroutine

Choose the user definition method to define the amplitude curve via coding in user subroutine **UAMP** (Abaqus/Standard) or **VUAMP** (Abaqus/Explicit). You define the value of the amplitude function in time and, optionally, the values of the derivatives and integrals for the function sought to be implemented as outlined in "UAMP," Section 1.1.19 of the Abaqus User Subroutines Reference Manual, and "VUAMP," Section 1.2.7 of the Abaqus User Subroutines Reference Manual.

You can use an arbitrary number of state variables that can be updated independently for each amplitude definition.

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

Moreover, solution-dependent sensors can be used to define the user-customized amplitude. The sensors can be identified via their name, and two utilities allow for the extraction of the current sensor value inside the user subroutine (see "Obtaining sensor information," Section 2.1.16 of the Abaqus User Subroutines Reference Manual). Simple control/logical models can be implemented using this feature as illustrated in "Crank mechanism," Section 4.1.2 of the Abaqus Example Problems Manual.

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

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

Using an amplitude definition with boundary conditions

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

When the displacement time history is defined by a piecewise linear amplitude variation (tabular or equally spaced amplitude definition), the corresponding velocity is piecewise constant and the acceleration may be infinite at the end of each time interval given in the amplitude definition table, as shown in Figure 32.1.2–8(a). This behavior is unreasonable. (In Abaqus/Explicit time derivatives of amplitude curves are typically based on finite differences, such as $\frac{[A(t_{i+1})-A(t_i)]}{\Delta t}$, so there is some inherent smoothing associated with the time discretization.)

You can modify the piecewise linear displacement variation into a combination of piecewise linear and piecewise quadratic variations through smoothing. Smoothing ensures that the velocity varies continuously during the time period of the amplitude definition and that the acceleration no longer has singularity points, as illustrated in Figure 32.1.2–8(b).

When the velocity time history is defined by a piecewise linear amplitude variation, the corresponding acceleration is piecewise constant. Smoothing can be used to modify the piecewise linear velocity variation into a combination of piecewise linear and piecewise quadratic variations. Smoothing ensures that the acceleration varies continuously during the time period of the amplitude definition.

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

In Abaqus/Explicit if a displacement jump is specified using an amplitude curve (i.e., the beginning displacement defined using the amplitude function does not correspond to the displacement at that time), this displacement jump will be ignored. Displacement boundary conditions are enforced in Abaqus/Explicit in an incremental manner using the slope of the amplitude curve. To avoid the "noisy" solution that may result in Abaqus/Explicit when smoothing is not used, it is better to specify the velocity history of a node rather than the displacement history (see "Boundary conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.3.1).

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

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

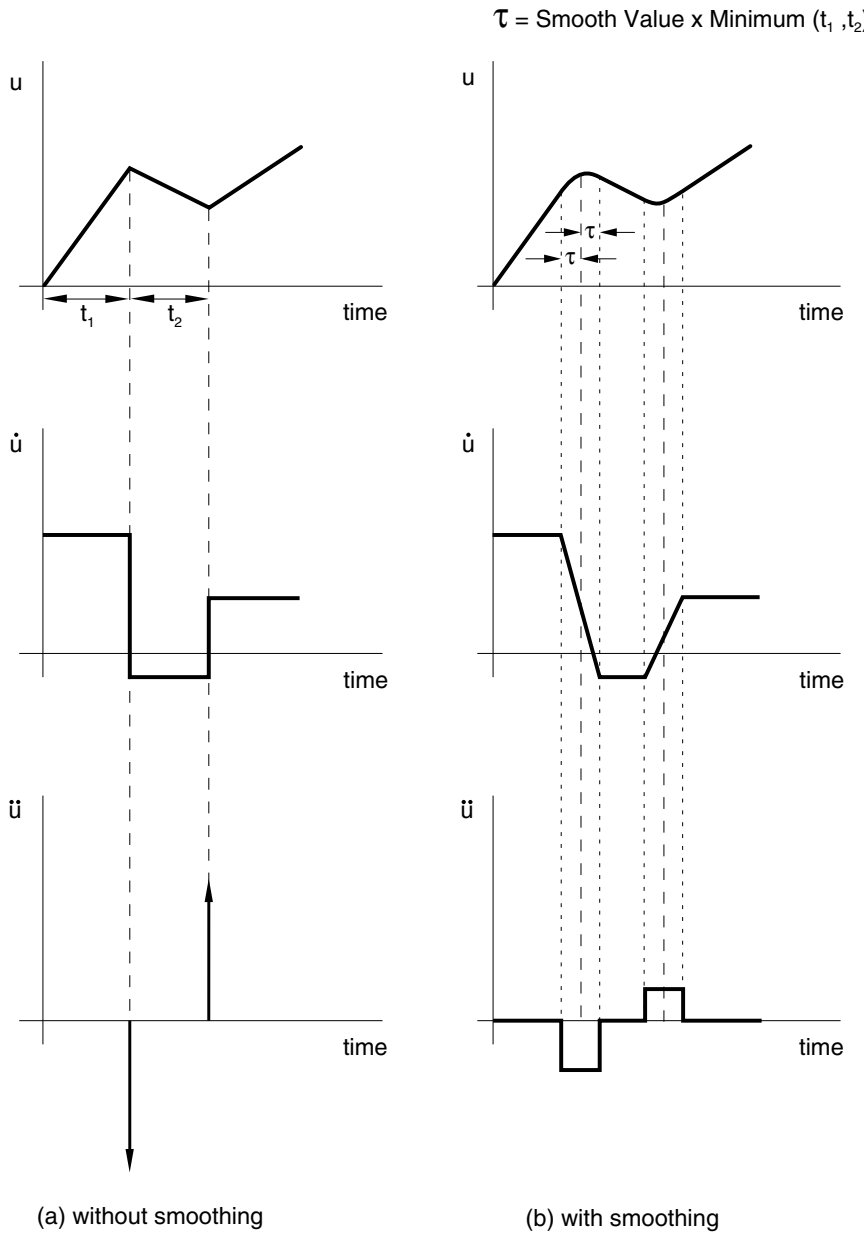


Figure 32.1.2–8 Piecewise linear displacement definitions.

Input File Usage: Use either of the following options:

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

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

Using an amplitude definition with secondary base motion in modal dynamics

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

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

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

Defining multiple amplitude curves

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

Scaling and shifting amplitude curves

You can scale and shift both time and magnitude when defining an amplitude. This can be helpful for example when your amplitude data need to be converted to a different unit system or when you reuse existing amplitude data to define similar amplitude curves. If both scaling and shifting are applied at the

same time, the amplitude values are first scaled and then shifted. The amplitude shifting and scaling can be applied to all amplitude definition types except for solution dependent, bubble, and user.

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

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

Reading the data from an alternate file

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

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

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

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

Baseline correction in Abaqus/Standard

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

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

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

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

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

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

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

Effects of baseline correction

The acceleration is modified by adding a quadratic variation of acceleration in time to the acceleration definition. The quadratic variation is chosen to minimize the mean squared velocity during each correction interval. Separate quadratic variations can be added for different correction intervals within

the amplitude definition by defining the correction intervals. Alternatively, the entire amplitude history can be used as a single correction interval.

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

The baseline correction technique is described in detail in “Baseline correction of accelerograms,” Section 6.1.2 of the Abaqus Theory Manual.

32.2 Initial conditions

- “Initial conditions in Abaqus/Standard and Abaqus/Explicit,” Section 32.2.1
- “Initial conditions in Abaqus/CFD,” Section 32.2.2

32.2.1 INITIAL CONDITIONS IN Abaqus/Standard AND Abaqus/Explicit

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Prescribed conditions: overview," Section 32.1.1
- *INITIAL CONDITIONS
- "Using the predefined field editors," Section 16.11 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Initial conditions are specified for particular nodes or elements, as appropriate. The data can be provided directly; in an external input file; or, in some cases, by a user subroutine or by the results or output database file from a previous Abaqus analysis.

If initial conditions are not specified, all initial conditions are zero except relative density in the porous metal plasticity model, which will have the value 1.0.

Specifying the type of initial condition being defined

Various types of initial conditions can be specified, depending on the analysis to be performed. Each type of initial condition is explained below, in alphabetical order.

Defining initial acoustic static pressure

In Abaqus/Explicit you can define initial acoustic static pressure values at the acoustic nodes. These values should correspond to static equilibrium and cannot be changed during the analysis. You can specify the initial acoustic static pressure at two reference locations in the model, and Abaqus/Explicit interpolates these data linearly to the acoustic nodes in the specified node set. The linear interpolation is based upon the projected position of each node onto the line defined by the two reference nodes. If the value at only one reference location is given, the initial acoustic static pressure is assumed to be uniform. The initial acoustic static pressure is used only in the evaluation of the cavitation condition (see "Acoustic medium," Section 25.3.1) when the acoustic medium is capable of undergoing cavitation.

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

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

Defining initial normalized concentration

In Abaqus/Standard you can define initial normalized concentration values for use with diffusion elements in mass diffusion analysis (see "Mass diffusion analysis," Section 6.9.1).

Input File Usage: *INITIAL CONDITIONS, TYPE=CONCENTRATION

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

Defining initially bonded contact surfaces

In Abaqus/Standard you can define initially bonded or partially bonded contact surfaces. This type of initial condition is intended for use with the crack propagation capability (see "Crack propagation analysis," Section 11.4.3). The surfaces specified have to be different; this type of initial condition cannot be used with self-contact.

If the crack propagation capability is not activated, the bonded portion of the surfaces will not separate. In this case defining initially bonded contact surfaces would have the same effect as defining tied contact, which generates a permanent bond between two surfaces during the entire analysis ("Defining tied contact in Abaqus/Standard," Section 34.3.7).

Input File Usage: *INITIAL CONDITIONS, TYPE=CONTACT

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

Define the initial location of an enriched feature

You can specify the initial location of an enriched feature, such as a crack, in an Abaqus/Standard analysis (see "Modeling discontinuities as an enriched feature using the extended finite element method," Section 10.7.1). Two signed distance functions per node are generally required to describe the crack location, including the location of crack tips, in a cracked geometry. The first signed distance function describes the crack surface, while the second is used to construct an orthogonal surface such that the intersection of the two surfaces defines the crack front. The first signed distance function is assigned only to nodes of elements intersected by the crack, while the second is assigned only to nodes of elements containing the crack tips. No explicit representation of the crack is needed because the crack is entirely described by the nodal data.

Input File Usage: *INITIAL CONDITIONS, TYPE=ENRICHMENT

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

Defining initial values of predefined field variables

You can define initial values of predefined field variables. The values can be changed during an analysis (see "Predefined fields," Section 32.6.1).

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

The definition of initial field variable values must be compatible with the section definition and with adjacent elements, as explained in "Predefined fields," Section 32.6.1.

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

Abaqus/CAE Usage: Initial predefined field variables are not supported in Abaqus/CAE.

Initializing predefined field variables with nodal temperature records from a user-specified results file

You can define initial values of predefined field variables using nodal temperature records from a particular step and increment of a results file from a previous Abaqus analysis or from a results file you create (see "Predefined fields," Section 32.6.1). The previous analysis is most commonly an Abaqus/Standard heat transfer analysis. The use of the **.fil** file extension is optional.

The part (**.prt**) file from the previous analysis is required to read the initial values of predefined field variables from the results file ("Defining an assembly," Section 2.10.1). Both the previous model and the current model must be consistently defined in terms of an assembly of part instances.

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

Abaqus/CAE Usage: Initial predefined field variables are not supported in Abaqus/CAE.

Defining initial predefined field variables using scalar nodal output from a user-specified output database file

You can define initial values of predefined field variables using scalar nodal output variables from a particular step and increment in the output database file of a previous Abaqus/Standard analysis. For a list of scalar nodal output variables that can be used to initialize a predefined field, see "Predefined fields," Section 32.6.1.

The part (**.prt**) file from the previous analysis is required to read initial values from the output database file (see "Defining an assembly," Section 2.10.1). Both the previous model and the current model must be defined consistently in terms of an assembly of part instances; node numbering must be the same, and part instance naming must be the same.

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

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

Abaqus/CAE Usage: Initial predefined field variables are not supported in Abaqus/CAE.

Defining initial predefined field variables by interpolating scalar nodal output variables for dissimilar meshes from a user-specified output database file

When the mesh for one analysis is different from the mesh for the subsequent analysis, Abaqus can interpolate scalar nodal output variables (using the undeformed mesh of the original analysis) to predefined field variables that you choose. For a list of supported scalar nodal output variables that can be used to define predefined field variables, see "Predefined fields," Section 32.6.1. This technique can also be used in cases where the meshes match but the node number or part instance naming differs between the analyses. Abaqus looks for the **.odb** extension automatically. The part (**.prt**) file from the previous analysis is required if that analysis model is defined in terms of an assembly of part instances (see "Defining an assembly," Section 2.10.1).

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

Abaqus/CAE Usage: Initial predefined field variables are not supported in Abaqus/CAE.

Defining initial fluid pressure in fluid-filled structures

You can prescribe initial pressure for fluid-filled structures (see "Surface-based fluid cavities: overview," Section 11.5.1).

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

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

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

Defining initial values of state variables for plastic hardening

You can prescribe initial equivalent plastic strain and, if relevant, the initial backstress tensor for elements that use one of the metal plasticity ("Inelastic behavior," Section 22.1.1) or Drucker-Prager ("Extended Drucker-Prager models," Section 22.3.1) material models. These initial quantities are intended for materials in a work hardened state; they can be defined directly or by user subroutine **HARDINI**. You can also prescribe initial values for the volumetric compacting plastic strain, $-\varepsilon_{\text{vol}}^{pl}$, for elements that use the crushable foam material model with volumetric hardening ("Crushable foam plasticity models," Section 22.3.5).

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

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

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

Defining hardening parameters for rebars

The hardening parameters can also be defined for rebars within elements. Rebars are discussed in "Defining rebar as an element property," Section 2.2.4.

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

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

Defining hardening parameters in user subroutine HARDINI

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

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

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

Defining elements initially open for tangential fluid flow

You can specify the pore pressure cohesive elements that are initially open for tangential fluid flow (see "Defining the constitutive response of fluid within the cohesive element gap," Section 31.5.7).

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

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

Defining initial mass flow rates in forced convection heat transfer elements

In Abaqus/Standard you can define the initial mass flow rate through forced convection heat transfer elements. You can specify a predefined mass flow rate field to vary the value of the mass flow rate within the analysis step (see "Uncoupled heat transfer analysis," Section 6.5.2).

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

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

Defining initial values of plastic strain

You can define an initial plastic strain field on elements that use one of the metal plasticity ("Inelastic behavior," Section 22.1.1) or Drucker-Prager ("Extended Drucker-Prager models," Section 22.3.1) material models. The specified plastic strain values will be applied uniformly over the element unless they are defined at each section point through the thickness in shell elements.

If a local coordinate system was defined (see "Orientations," Section 2.2.5), the plastic strain components must be given in the local system.

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

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

Defining initial plastic strains for rebars

Initial values of stress can also be defined for rebars within elements (see "Defining rebar as an element property," Section 2.2.4).

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

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

Defining initial pore fluid pressures in a porous medium

In Abaqus/Standard you can define the initial pore pressure, u_w, for nodes in a coupled pore fluid diffusion/stress analysis (see "Coupled pore fluid diffusion and stress analysis," Section 6.8.1). The initial pore pressure can be defined either directly as an elevation-dependent function or by user subroutine **UPOREP**.

Elevation-dependent initial pore pressures

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

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

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

Defining initial pore pressures in user subroutine `UPOREP`

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

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

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

Defining initial pore pressure values using nodal pore pressure output from a user-specified output database file

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

For the same mesh pore pressure mapping, both the previous model and the current model must be defined consistently, including node numbering, which must be the same in both models. If the models are defined in terms of an assembly of part instances, the part instance naming must be the same.

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

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

Interpolating initial pore pressure values for dissimilar pore pressure mapping values in a user-specified output database file

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

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

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

Defining initial pressure stress in a mass diffusion analysis

In Abaqus/Standard you can specify the initial pressure stress, $p \overset{\text{def}}{=} -\text{trace}(\boldsymbol{\sigma})/3$, at the nodes in a mass diffusion analysis (see "Mass diffusion analysis," Section 6.9.1).

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

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

Defining initial pressure stress from a user-specified results file

You can define initial values of pressure stress as those values existing at a particular step and increment in the results file of a previous Abaqus/Standard stress/displacement analysis (see "Predefined fields," Section 32.6.1). The use of the `.fil` file extension is optional. The initial values of pressure stress cannot be read from the results file when the previous model or the current model is defined in terms of an assembly of part instances ("Defining an assembly," Section 2.10.1).

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

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

Defining initial void ratios in a porous medium

In Abaqus/Standard you can specify the initial values of the void ratio, e, at the nodes of a porous medium (see "Coupled pore fluid diffusion and stress analysis," Section 6.8.1). The initial void ratio can be defined either directly as an elevation-dependent function, by interpolation from a previous output database file, or by user subroutine `VOIDRI`.

Elevation-dependent initial void ratio

When an elevation-dependent void ratio is prescribed for a particular node set, the void ratio in the vertical direction (assumed to be the z-direction in three-dimensional and axisymmetric models and the y-direction in two-dimensional models) is assumed to vary linearly with this vertical coordinate. When the void ratio is specified for a region meshed with fully integrated first-order elements, the nodal values

of void ratio are interpolated to the centroid of the element and are assumed to be constant through the element. You must provide two pairs of void ratio and elevation values to define the void ratio throughout the node set. Enter only the first void ratio value (omit the second void ratio value and the elevation values) to define a constant void ratio distribution.

Input File Usage: *INITIAL CONDITIONS, TYPE=RATIO

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

Defining void ratio from a user-specified output database

You can define initial void ratios from the output database (**`.odb`**) file of a previous Abaqus/Standard soil analysis in which the void ratio is requested as output.

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

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

Interpolating initial void ratios from values in a user-specified output database

When you define initial void ratios from the output database (**`.odb`**) file of a previous Abaqus/Standard soil analysis, you can also limit the interpolation region by specifying the source region in the form of an element set from which void ratios are to be interpolated and the target region in the form of a node set onto which the void ratios are mapped.

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

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

Defining void ratios in user subroutine `VOIDRI`

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

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

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

Defining a reference mesh for membrane elements

In Abaqus/Explicit you can specify a reference mesh (initial metric) for membrane elements. This is typically useful in finite element airbag simulations to model the wrinkles that arise from the airbag folding process. A flat mesh may be suitable for the unstressed reference configuration, but the initial state may require a corresponding folded mesh defining the folded state. Defining a reference configuration that is different from the initial configuration may result in nonzero stresses and strains in the initial configuration based on the material definition. If a reference mesh is specified for an element, any initial stress or strain conditions specified for the same element are ignored.

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

You can define the reference mesh using either the element numbers and the coordinates of the nodes in each element or the node numbers and the coordinates of the nodes. The coordinates of all of the nodes in the element have to be specified for both methods to have a valid initial condition for that element. The two alternatives are mutually exclusive.

Input File Usage: Specifying the reference mesh using element numbers and coordinates of all of the element's nodes:

*INITIAL CONDITIONS, TYPE=REF COORDINATE

Specifying the reference mesh using node numbers and the coordinates of the nodes:

*INITIAL CONDITIONS, TYPE=NODE REF COORDINATE

Abaqus/CAE Usage: The specification of a reference mesh for membrane elements is not supported in Abaqus/CAE.

Defining initial relative density

You can specify the initial values of the relative density field for a porous metal plasticity material model (see "Porous metal plasticity," Section 22.2.9) or equations of state (see "Equation of state," Section 24.2.1).

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

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

Defining initial angular and translational velocity

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

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

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

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

Defining initial saturation for a porous medium

In Abaqus/Standard you can define the initial saturation, s, for elements in a coupled pore fluid diffusion/stress analysis (see "Coupled pore fluid diffusion and stress analysis," Section 6.8.1).

Input File Usage: *INITIAL CONDITIONS, TYPE=SATURATION

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

Defining the initial values of solution-dependent state variables

You can define initial values of solution-dependent state variables (see "User subroutines: overview," Section 17.1.1). The initial values can be defined directly or, in Abaqus/Standard, by user subroutine **SDVINI**. Values given directly will be applied uniformly over the element.

Input File Usage: *INITIAL CONDITIONS, TYPE=SOLUTION

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

Defining the initial values of solution-dependent state variables for rebars

The initial values of solution-dependent variables can also be defined for rebars within elements. Rebars are discussed in "Defining rebar as an element property," Section 2.2.4.

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

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

Defining the initial values of solution-dependent state variables in user subroutine SDVINI

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

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

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

Defining initial specific energy for equations of state

In Abaqus/Explicit you can specify the initial values of the specific energy for equations of state (see "Equation of state," Section 24.2.1).

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

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

Defining spud can embedment or spud can preload

In Abaqus/Standard you can define an initial embedment of a spud can. Alternatively, you can define an initial vertical preload of a spud can (see "Elastic-plastic joints," Section 31.10.1).

Input File Usage: Use one of the following options:

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

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

Defining initial stresses

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

If a local coordinate system was defined (see "Orientations," Section 2.2.5), stresses must be given in the local system.

In soils (porous medium) problems the initial effective stress should be given; see "Coupled pore fluid diffusion and stress analysis," Section 6.8.1, for a discussion of defining initial conditions in porous media.

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

Initial stress fields cannot be defined for spring elements. See "Springs," Section 31.1.1, for a discussion of defining initial forces in spring elements.

Initial stress fields cannot be defined for elements using a fabric material. However, an initial stress and strain state can be introduced in a fabric material made of membrane elements by defining a reference mesh (see "Defining a reference mesh for membrane elements " above).

Input File Usage: *INITIAL CONDITIONS, TYPE=STRESS

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

Defining initial stresses for rebars

Initial values of stress can also be defined for rebars within elements (see "Defining rebar as an element property," Section 2.2.4).

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

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

Defining initial stresses that vary through the thickness of shell elements

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

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

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

Defining initial stresses in user subroutine `SIGINI`

For complicated cases (such as elbow elements) in Abaqus/Standard the initial stress field can be defined by user subroutine **`SIGINI`**. In this case Abaqus/Standard will make a call to subroutine **`SIGINI`** at the start of the analysis for each material calculation point in the model. You can then define all active stress components at each point as functions of coordinates, element number, etc.

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

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

Defining initial stresses using stress output from a user-specified output database file

You can define initial stresses using stress output variables from a particular step and increment in the output database (**`.odb`**) file of a previous Abaqus/Standard analysis.

In this case both the previous model and the current model must be defined consistently. The element numbering and element types must be the same in both models. If the models are defined in terms of an assembly of part instances, part instance naming must be the same.

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

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

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

Establishing equilibrium in Abaqus/Standard

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

In a soils analysis (that is, for models containing elements that include pore fluid pressure as a variable) the geostatic stress field procedure ("Geostatic stress state," Section 6.8.2) should be used for the equilibrating step. Any initial loading (such as geostatic gravity loads) that contributes to the initial equilibrium should be included in this step definition. The initial time increment and the total time specified in this step should be the same. The initial stresses are applied in full at time zero; and if equilibrium can be achieved, this step will converge in one increment. Therefore, there is no benefit to incrementing.

To achieve equilibrium for all other analyses, a first step using the static procedure ("Static stress analysis," Section 6.2.2) should be used. It is recommended that you specify the initial time increment to be equal to the total time specified in this step so that Abaqus/Standard will attempt to find equilibrium in one increment. By default, Abaqus/Standard ramps down the unbalanced stress over the first step. This allows Abaqus/Standard to use automatic incrementation if equilibrium cannot be found in one increment. This ramping is achieved in the following manner:

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

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

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

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

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

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

Establishing equilibrium in Abaqus/Explicit

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

Defining elevation-dependent (geostatic) initial stresses

You can define elevation-dependent initial stresses. When a geostatic stress state is prescribed for a particular element set, the stress in the vertical direction (assumed to be the z-direction in

three-dimensional and axisymmetric models and the y-direction in two-dimensional models) is assumed to vary (piecewise) linearly with this vertical coordinate.

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

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

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

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

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

Defining initial temperatures

You can define initial temperatures at the nodes of either heat transfer or stress/displacement elements. The temperatures of stress/displacement elements can be changed during an analysis (see "Predefined fields," Section 32.6.1).

The definition of initial temperature values must be compatible with the section definition of the element and with adjacent elements, as explained in "Predefined fields," Section 32.6.1.

Input File Usage: *INITIAL CONDITIONS, TYPE=TEMPERATURE

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

Defining initial temperatures from a user-specified results or output database file

You can define initial temperatures as those values existing as nodal temperatures at a particular step and increment in the results or output database file of a previous Abaqus/Standard heat transfer analysis (see "Predefined fields," Section 32.6.1).

The part (**`.prt`**) file from the previous analysis is required to read initial temperatures from the results or output database file (see "Defining an assembly," Section 2.10.1). Both the previous model and the current model must be consistently defined in terms of an assembly of part instances; node numbering must be the same, and part instance naming must be the same.

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

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

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

Interpolating initial temperatures for dissimilar meshes from a user-specified results or output database file

When the mesh for the heat transfer analysis is different from the mesh for the subsequent stress/displacement analysis, Abaqus can interpolate the temperature values from the nodes in the undeformed heat transfer model to the current nodal temperatures. This technique can also be used in cases where the meshes match but the node number or part instance naming differs between the analyses. Only temperatures from an output database file can be used for the interpolation; Abaqus will look for the `.odb` extension automatically. The part (`.prt`) file from the previous analysis is required if that analysis model is defined in terms of an assembly of part instances (see "Defining an assembly," Section 2.10.1).

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

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

Interpolating initial temperatures for dissimilar meshes with user-specified regions

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

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

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

Interpolating initial temperatures for meshes that differ only in element order from a user-specified results or output database file

If the only difference in the meshes is the element order (first-order elements in the heat transfer model and second-order elements in the stress/displacement model), in Abaqus/Standard you can indicate that midside node temperatures in second-order elements are to be interpolated from corner node temperatures read from the results or output database file of the previous heat transfer analysis using first-order elements. You must ensure that the corner node temperatures are not defined using a mixture of direct data input and reading from the results or output database file, since midside node temperatures that give unrealistic temperature fields may result. In practice, the capability for calculating midside node temperatures is most useful when temperatures generated by a heat transfer analysis are read from the results or output database file for the whole mesh during the stress analysis. Once the midside node capability is activated, the capability will remain active for the rest of the analysis, including for any predefined temperature fields defined to change temperatures during the analysis. The general interpolation and midside node capabilities are mutually exclusive.

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

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

Defining initial velocities for specified degrees of freedom

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

Initial velocities must be defined in global directions, regardless of the use of local transformations (“Transformed coordinate systems,” Section 2.1.5).

Input File Usage: *INITIAL CONDITIONS, TYPE=VELOCITY

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

Defining initial volume fractions for Eulerian elements

You can define initial volume fractions to create material within Eulerian elements in Abaqus/Explicit. By default, these elements are filled with void. See “Initial conditions” in “Eulerian analysis,” Section 14.1.1, for a description of strategies for initializing Eulerian materials.

Input File Usage: *INITIAL CONDITIONS, TYPE=VOLUME FRACTION

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

Reading the input data from an external file

The input data for an initial conditions definition can be contained in a separate file. See "Input syntax rules," Section 1.2.1, for the syntax of such file names.

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

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

Consistency with kinematic constraints

Abaqus does not ensure that initial conditions are consistent with multi-point or equation constraints for nodal quantities other than velocity (see "General multi-point constraints," Section 33.2.2, and "Linear constraint equations," Section 33.2.1). Initial conditions on nodal quantities such as temperature in heat transfer analysis, pore pressure in soils analysis, or acoustic pressure in acoustic analysis must be prescribed to be consistent with any multi-point constraint or equation constraint governing these quantities.

Spatial interpolation method

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

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

32.2.2 INITIAL CONDITIONS IN Abaqus/CFD

Products: Abaqus/CFD Abaqus/CAE

References

- "Prescribed conditions: overview," Section 32.1.1
- "Using the predefined field editors," Section 16.11 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

In Abaqus/CFD initial conditions for fluid flow simulation are specified using element sets.

Defining initial velocities

You can define the initial fluid flow velocity in elements; however, if such conditions are omitted, a default value of zero is assumed. Initial velocities must be defined in global directions, regardless of the use of local transformations (see "Transformed coordinate systems," Section 2.1.5).

For incompressible flow Abaqus/CFD automatically uses the user-defined boundary conditions and tests the specified initial velocity to be sure that the initial velocity field is divergence-free and that the velocity boundary conditions are compatible with the initial velocity field. If they are not, the initial velocity is projected onto a divergence-free subspace, yielding initial conditions that define a well-posed incompressible Navier-Stokes problem. Therefore, in some circumstances, the user-specified initial velocity may be overridden with a velocity that is divergence-free and matches the velocity boundary conditions.

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

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: **Category: Fluid**: **Fluid velocity**

Defining initial density

You can define the initial fluid density in elements. However, if the initial condition is omitted, the material density definition is assumed as default (see "Density," Section 20.2.1). Similarly, if the initial density is specified on an element set that does not include all fluid elements, the material density is assumed as the default for those elements not contained in the element set.

Input File Usage: *INITIAL CONDITIONS, TYPE=DENSITY, ELEMENT AVERAGE

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: **Category: Fluid**: **Fluid density**

Initial pressure for incompressible fluid flow

For incompressible flows it is not necessary to prescribe the initial pressure condition since the initial pressure field is computed automatically from the initial velocity field and boundary conditions. This is done to ensure proper starting conditions for incompressible flows.

Defining initial temperature

If the energy equation is solved, the initial fluid temperature in elements must be defined.

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

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: **Category: Fluid**: **Fluid thermal energy**

Defining initial Spalart-Allmaras turbulent eddy viscosity for fluid flow

If the Spalart-Allmaras turbulence model is active, you must prescribe an initial value for the Spalart-Allmaras turbulent eddy viscosity that is greater than zero and roughly three to five times the kinematic viscosity. The kinematic viscosity is the ratio of the fluid viscosity and density ($\nu = \eta/\rho$). For more information, see "Viscosity," Section 25.1.4.

Input File Usage: *INITIAL CONDITIONS, TYPE=TURBNU, ELEMENT AVERAGE

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: **Category: Fluid**: **Fluid turbulence**; **Eddy viscosity**: $\widetilde{\nu}$

Defining initial k and ε for fluid flow

If the RNG k–ε turbulence model is active, initial conditions need to be specified for both k and ε. The k and ε values must be greater than zero. A simple procedure to approximate the initial conditions can be obtained from values of the turbulence intensity and an approximate initial turbulent eddy viscosity as described below.

The turbulent kinetic energy is defined as

$$k = \frac{3u'^2}{2},$$

where u' is the characteristic velocity scale or root mean square velocity that is usually related to the characteristic velocity scale of the flow (U_∞) through the turbulence intensity,

$$I = \frac{u'}{U_\infty}.$$

Therefore, an estimation for the initial conditions for the turbulent kinetic energy, k, can be expressed in terms of the characteristic velocity and turbulence intensity as

$$k = \frac{3U_\infty^2 I^2}{2}.$$

The initial value for the turbulent kinetic energy dissipation, ε, can be obtained from a known/proposed level of the turbulent eddy viscosity, ν_T, as

$$\varepsilon = C_\mu \frac{k^2}{\nu_T},$$

where C_μ is the k–ε turbulent viscosity model coefficient and ν is the fluid kinematic viscosity.

Input File Usage: Use the following option to specify the initial turbulent kinetic energy:

∗INITIAL CONDITIONS, TYPE=TURBKE, ELEMENT AVERAGE

Use the following option to specify the initial turbulent kinetic energy dissipation rate:

∗INITIAL CONDITIONS, TYPE=TURBEPS, ELEMENT AVERAGE

Abaqus/CAE Usage: Load module: **Create Predefined Field**: **Step: Initial**: **Category: Fluid**: **Fluid turbulence**; **Turbulent kinetic energy**: k, **Dissipation rate**: ε

32.3 Boundary conditions

- “Boundary conditions in Abaqus/Standard and Abaqus/Explicit,” Section 32.3.1
- “Boundary conditions in Abaqus/CFD,” Section 32.3.2

32.3.1 BOUNDARY CONDITIONS IN Abaqus/Standard AND Abaqus/Explicit

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Defining a model in Abaqus," Section 1.3.1
- "Prescribed conditions: overview," Section 32.1.1
- "VDISP," Section 1.2.1 of the Abaqus User Subroutines Reference Manual
- "DISP," Section 1.1.4 of the Abaqus User Subroutines Reference Manual
- *BOUNDARY
- "Using the boundary condition editors," Section 16.10 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Boundary conditions:

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

Relative motions in connector elements can be prescribed similar to boundary conditions. See "Connector actuation," Section 30.1.3, for more detailed information.

Prescribing boundary conditions as model data

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

For both "direct" and "type" format you specify the region of the model to which the boundary conditions apply and the degrees of freedom to be restrained. (See "Conventions," Section 1.2.2, for the degree of freedom numbers used in Abaqus.)

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

Input File Usage: *BOUNDARY

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

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

Using the direct format

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

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

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

For example,

```
*BOUNDARY
EDGE, 1
```

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

```
EDGE, 1, 4
```

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

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

Use one of the following options:

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

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

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

Using the "type" format in stress/displacement analyses

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

XSYMM	Symmetry about a plane $X = \text{constant}$ (degrees of freedom $1, 5, 6 = 0$).
YSYMM	Symmetry about a plane $Y = \text{constant}$ (degrees of freedom $2, 4, 6 = 0$).

ZSYMM	Symmetry about a plane $Z = \text{constant}$ (degrees of freedom $3, 4, 5 = 0$).
ENCASTRE	Fully built-in (degrees of freedom $1, 2, 3, 4, 5, 6 = 0$).
PINNED	Pinned (degrees of freedom $1, 2, 3 = 0$).

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

XASYMM	Antisymmetry about a plane with $X = \text{constant}$ (degrees of freedom 2, 3, $4 = 0$).
YASYMM	Antisymmetry about a plane with $Y = \text{constant}$ (degrees of freedom 1, 3, $5 = 0$).
ZASYMM	Antisymmetry about a plane with $Z = \text{constant}$ (degrees of freedom 1, 2, $6 = 0$).

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

NOWARP	Prevent warping of an elbow section at a node.
NOOVAL	Prevent ovalization of an elbow section at a node.
NODEFORM	Prevent all cross-sectional deformation (warping, ovalization, and uniform radial expansion) at a node.

The NOWARP, NOOVAL, and NODEFORM types apply only to elbow elements ("Pipes and pipebends with deforming cross-sections: elbow elements," Section 28.5.1).

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

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

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

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

Prescribing boundary conditions as history data

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

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

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

Input File Usage: *BOUNDARY

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

Using the direct format

Specify the region of the model to which the boundary conditions apply, the degree or degrees of freedom to be specified (see "Conventions," Section 1.2.2, for the degree of freedom numbers used in Abaqus), and the magnitude of the boundary condition. If the magnitude is omitted, it is the same as specifying a zero magnitude.

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

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

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

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

*BOUNDARY, TYPE=VELOCITY

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

*BOUNDARY, TYPE=ACCELERATION

For example,

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

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

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

Select one of the following categories and types:

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

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

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

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

Prescribed displacement

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

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

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

Using the "type" format in stress/displacement analyses

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

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

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

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

Defining boundary conditions that vary with time

The prescribed magnitude of a basic solution variable, a velocity, or an acceleration can vary with time during a step according to an amplitude definition ("Amplitude curves," Section 32.1.2).

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

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

Abaqus/Explicit does not apply default smoothing to discontinuous amplitude curves. To avoid the "noisy" solution that may result from discontinuities in Abaqus/Explicit, it is better to specify the velocity history of a node. See "Amplitude curves," Section 32.1.2.

Input File Usage: Use both of the following options:

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

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

Defining boundary condition through user subroutines

If an amplitude based evolution of a boundary condition is not sufficient, you can define it yourself in a user subroutine. For this purpose, Abaqus/Standard provides the routine **DISP**; whereas, Abaqus/Explicit provides the routine **VDISP**. The region to which the boundary conditions apply and the constrained degrees of freedom are specified as part of the boundary condition definition. The actual boundary condition is set within the user routine based on a number of variables made available in those routines (see "DISP," Section 1.1.4 of the Abaqus User Subroutines Reference Manual for **DISP** and "VDISP," Section 1.2.1 of the Abaqus User Subroutines Reference Manual for **VDISP**).

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

Input File Usage: ∗BOUNDARY, USER

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

Boundary condition propagation

By default, all boundary conditions defined in the previous general analysis step remain unchanged in the subsequent general step or in subsequent consecutive linear perturbation steps. Boundary conditions do not propagate between linear perturbation steps. You define the boundary conditions in effect for a given step relative to the preexisting boundary conditions. At each new step the existing boundary conditions can be modified and additional boundary conditions can be specified. Alternatively, you can release all previously applied boundary conditions in a step and specify new ones. In this case any boundary conditions that are to be retained must be respecified.

Modifying boundary conditions

When you modify an existing boundary condition, the node or node set must be specified in exactly the same way as previously. For example, if a boundary condition is specified for a node set in one step and

for an individual node contained in the set in another step, Abaqus issues an error. You must remove the boundary condition and respecify it to change the way the node or node set is specified.

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

*BOUNDARY
*BOUNDARY, OP=MOD

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

Removing boundary conditions

If you choose to remove any boundary condition in a step, no boundary conditions will be propagated from the previous general step. Therefore, all boundary conditions that are in effect during this step must be respecified. The only exception to this rule is during an eigenvalue buckling prediction procedure, as described in "Eigenvalue buckling prediction," Section 6.2.3.

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

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

*BOUNDARY, OP=NEW

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

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

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

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

Fixing degrees of freedom at a point in an Abaqus/Standard analysis

In Abaqus/Standard you can "freeze" specified degrees of freedom at their final values from the last general analysis step. Specifying a zero velocity or zero acceleration boundary condition will have the same effect as fixing the degrees of freedom for displacement or velocity, respectively.

Input File Usage: *BOUNDARY, FIXED

The OP=NEW parameter must be used with the FIXED parameter if there are any other *BOUNDARY options in the same step that have the OP=NEW parameter. Any magnitudes given for the boundary condition are ignored.

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

Prescribing boundary conditions in linear perturbation steps

In a linear perturbation step ("General and linear perturbation procedures," Section 6.1.2) the magnitudes of prescribed boundary conditions should be given as the magnitudes of the perturbations about the base state. Boundary conditions given within the model definition are always regarded as part of the base state, even if the first analysis step is a linear perturbation step. The boundary conditions given in a linear perturbation step will not affect subsequent steps.

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

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

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

The antisymmetric buckling modes of a symmetric structure can be found in an eigenvalue buckling prediction analysis by specifying the proper boundary conditions (see "Eigenvalue buckling prediction," Section 6.2.3).

Prescribing real and imaginary values in boundary conditions

In steady-state dynamic and matrix generation procedures, a boundary condition can be prescribed using either a real or an imaginary value (see "Direct-solution steady-state dynamic analysis," Section 6.3.4, and "Generating matrices," Section 10.3.1). If the real value is prescribed for a degree of freedom (and the imaginary value is not explicitly prescribed), the imaginary value is considered to be zero. Similarly, if the imaginary value is prescribed (and the real value is not explicitly prescribed), the real value is considered to be zero.

Prescribed motion in modal superposition procedures

In modal superposition procedures ("Dynamic analysis procedures: overview," Section 6.3.1) prescribed displacements cannot be defined directly using a boundary condition. Instead, the boundary conditions are grouped into bases in a frequency extraction step. Then, the motion of each base is prescribed in the modal superposition step. See "Natural frequency extraction," Section 6.3.5, and "Transient modal dynamic analysis," Section 6.3.7, for details on this method.

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

Abaqus/CAE Usage: Base motions are not supported in Abaqus/CAE.

Submodeling

When using the submodeling technique, the magnitudes of the boundary conditions in the submodel can be defined by interpolating the values of the prescribed degrees of freedom from the file output results of the global model. See "Node-based submodeling," Section 10.2.2, for details.

Prescribing large rotations

Sequential finite rotations about different axes of rotation are not additive, which can make direct specification of such rotations challenging. It is much simpler to apply finite-rotation boundary conditions by specifying the rotational velocity versus time. For a discussion of the rotation degrees of freedom and a multiple step finite rotation example that demonstrates why velocity-type boundary conditions are preferred for specifying finite-rotation boundary conditions, see "Conventions," Section 1.2.2.

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

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

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

Example: Using velocity-type boundary conditions to prescribe rotations

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

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

If, in the next step, the same node should have an additional rotation of $\pi/2$ radians about the global x-axis, use another static step with a step time of 1.0 and again define a velocity-type boundary condition

to prescribe zero velocity for degrees of freedom 5 and 6 and a constant angular velocity of $\pi/2$ for degree of freedom 4.

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

32.3.2 BOUNDARY CONDITIONS IN Abaqus/CFD

Products: Abaqus/CFD Abaqus/CAE

References

- "Prescribed conditions: overview," Section 32.1.1
- "Conventions," Section 1.2.2
- *BOUNDARY
- *FLUID BOUNDARY
- "Using the boundary condition editors," Section 16.10 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Boundary conditions:

- are used to prescribe the values of all primitive variables involved in a fluid dynamics calculation (e.g., velocities, temperatures, turbulence variables, wall-normal distance, etc.);
- can be given as "history" input data (within an analysis step) to add, modify, or remove zero-valued or nonzero boundary conditions; and
- can be prescribed through the use of a co-simulation region for multiphysics problems.

Computational fluid dynamics problems typically require the prescription of multiple variables such as pressure, temperature, and velocity for boundary conditions. In practice, boundary conditions tend to appear together to collectively define a physical behavior; e.g., no-slip/no-penetration conditions at a wall. In contrast, Neumann conditions (e.g., prescribed heat flux) are specified as loads (see "Specifying surface-based distributed heat fluxes" in "Thermal loads," Section 32.4.4). In the absence of a prescribed boundary condition or load, the default behavior for Abaqus/CFD is to enforce a homogeneous (zero) Neumann condition. For example, if the temperature is not specified at a wall, the default behavior is to automatically specify a perfectly insulated boundary; i.e., zero normal heat flux. Similarly, if the velocity is not prescribed, the normal derivative of the velocity is set to zero.

In Abaqus/CFD combinations of boundary conditions that represent an inflow, outflow, or wall behavior are grouped collectively for ease of use. These types of boundary conditions are prescribed using Abaqus/CAE (for more information, see "Using the boundary condition editors," Section 16.10 of the Abaqus/CAE User's Manual).

Active degrees of freedom

In Abaqus/CFD the active fields (degrees of freedom) are determined by the analysis procedure and the options specified, such as turbulence models and auxiliary transport equations. You specify a boundary condition type to identify the degree of freedom for a fluid boundary condition. Element-based and

node-based degrees of freedom and the analysis procedure and additional options required for activation, if any, are listed in Table 32.3.2–1 and Table 32.3.2–2, respectively.

Table 32.3.2–1 Element-based degrees of freedom and activation options for fluid boundary conditions.

Boundary condition type	Description	Incompressible flow
TEMP	Fluid temperature	Energy equation
TEMP*n*	Fluid temperature on face *n*	Energy equation
TURBEPS	Turbulent energy dissipation rate (ε)	RNG k-ε model
TURBEPS*n*	Turbulent energy dissipation rate (ε) on face *n*	RNG k-ε model
TURBKE	Turbulent kinetic energy (k)	RNG k-ε model
TURBKE*n*	Turbulent kinetic energy (k) on face *n*	RNG k-ε model
TURBNU	Turbulent kinematic eddy viscosity	Spalart-Allmaras model
TURBNU*n*	Turbulent kinematic eddy viscosity on face *n*	Spalart-Allmaras model
VELX	x-velocity	—
VELX*n*	x-velocity on face *n*	—
VELY	y-velocity	—
VELY*n*	y-velocity on face *n*	—
VELZ	z-velocity	—
VELZ*n*	z-velocity on face *n*	—

Table 32.3.2–2 Node-based degrees of freedom and activation options for fluid boundary conditions.

Boundary condition type	Description	Incompressible flow
P	Fluid pressure	—
DIST	Wall-distance normal function	—

Prescribing inflow and outflow boundary conditions

You can specify boundary conditions to describe the flow behavior where fluid enters the analysis domain and where the fluid leaves the analysis domain.

Input File Usage: Use the following option to define the inflow and outflow boundary conditions at surfaces:

∗FLUID BOUNDARY, TYPE=SURFACE
surface name, *boundary condition type label*, *magnitude*

where *boundary condition type label* is VELX, VELY, VELZ, TEMP, TURBKE, TURBEPS, TURBNU, or P.

Use the following option to define distributed inflow and outflow boundary conditions at element faces:

∗FLUID BOUNDARY, TYPE=ELEMENT
element set label, *boundary condition type label*, *magnitude*

where *boundary condition type label* is VELX*n*, VELY*n*, VELZ*n*, TEMP*n*, TURBKE*n*, TURBEPS*n*, or TURBNU*n*.

Use the following option to define distributed inflow and outflow boundary conditions at nodes:

∗FLUID BOUNDARY, TYPE=NODE
node set label, P, *magnitude*

Abaqus/CAE Usage: Use the following option to define the inflow and outflow boundary conditions at surfaces:

Load module: **Create Boundary Condition**: **Step**: ***flow_step***: **Category**: **Fluid**: **Fluid inlet/outlet**: select inlet regions or outlet regions; and specify momentum (pressure or velocity), thermal energy (temperature), and turbulence conditions at the inlet or outlet

Defining distributed inflow and outflow boundary conditions at element faces is supported in Abaqus/CAE only for velocity boundary conditions.

Use the following option:

Load module: **Create Boundary Condition**: **Step**: ***flow_step***: **Category**: **Fluid**: **Fluid inlet/outlet**: select inlet regions or outlet regions; **Momentum**: toggle on **Specify**, and choose **Velocity**; **Distribution**: select an analytical field

Defining distributed inflow and outflow boundary conditions at nodes is not supported in Abaqus/CAE.

Inflow boundary conditions

An inflow boundary condition is used to describe the flow behavior at a surface where fluid enters the analysis domain. For incompressible flows, inflow conditions can be prescribed for velocity or pressure, temperature, and turbulence variables. If boundary conditions are not specified explicitly for a variable, a homogeneous Neumann condition is assumed automatically. This corresponds to permitting the variable (e.g., temperature) to vary at the inflow and the incoming fluid to correspond to that local variable. Similarly, if pressure is not specified, its normal derivative at the inflow surface is automatically set to zero. The velocity components can be prescribed independently.

Outflow boundary conditions

An outflow boundary corresponds to a surface where the fluid flow leaves the analysis domain. In Abaqus/CFD outflow conditions are most frequently associated with a specified pressure. However, all other flow variables can be prescribed at an outflow boundary as well. Similar to an inflow boundary, when a variable is not specified, its normal derivative is assumed to be zero. As such, convective outflows carry their quantities out of the domain at a fixed level, resulting in essentially nonreflecting boundaries.

Prescribing wall boundary conditions

Wall boundary conditions are typically associated with the no-slip/no-penetration behavior at a solid surface. However, the behavior at a solid wall may also require the prescription of temperature and, optionally, turbulence variables depending on the flow conditions. In situations where a wall heat flux is required, a heat flux loading must be prescribed in addition to the wall boundary conditions.

Depending on the physical properties of the wall, the wall boundary conditions can be modified to achieve a variety of physical behaviors that include slip, no-slip, infiltration, symmetry, etc.

Input File Usage: Use the following option to define wall boundary conditions at surfaces:

∗FLUID BOUNDARY, TYPE=SURFACE
surface name, *boundary condition type label*, *magnitude*

where *boundary condition type label* is VELX, VELY, VELZ, TEMP, TURBKE, TURBEPS, TURBNU, P or DIST.

Use the following option to define distributed wall boundary conditions at element faces:

∗FLUID BOUNDARY, TYPE=ELEMENT
element set label, *boundary condition type label*, *magnitude*

where *boundary condition type label* is VELX*n*, VELY*n*, VELZ*n*, TEMP*n*, TURBKE*n*, TURBEPS*n*, or TURBNU*n*.

Use the following option to define distributed wall boundary conditions at nodes:

*FLUID BOUNDARY, TYPE=NODE
node set label, P, *magnitude*

For example, use the following settings for a no-slip/no-penetration wall that is not moving and with the Spalart-Allmaras turbulence model active (wall-normal distance boundary condition and turbulent eddy viscosity set to zero at the wall):

```
*FLUID BOUNDARY, TYPE=SURFACE
surface name, DIST, 0
surface name, VELX, 0
surface name, VELY, 0
surface name, VELZ, 0
surface name, TURBNU, 0
```

Abaqus/CAE Usage: Use the following option to define wall boundary conditions at surfaces:

Load module: **Create Boundary Condition**: **Step**: ***flow_step***: **Category**: **Fluid**: **Fluid wall condition**: select regions; select **Condition**: **No slip**, **Shear**, or **Infiltration**; and specify velocity, thermal energy (temperature), and turbulence conditions at the wall

Defining distributed wall boundary conditions at elements is supported in Abaqus/CAE only for velocity boundary conditions at a slip wall or infiltration wall.

Use the following option to define distributed wall boundary conditions at elements:

Load module: **Create Boundary Condition**: **Step**: ***flow_step***: **Category**: **Fluid**: **Fluid wall condition**: select regions; **Velocity**: **Distribution**: select an analytical field

Defining distributed wall boundary conditions at nodes is not supported in Abaqus/CAE.

No-slip/no-penetration wall

A no-slip (and no-penetration) wall is a surface where the fluid adheres to the wall without penetrating it. No-slip/no-penetration conditions are prescribed by setting all velocity components equal to the wall velocity (zero if the wall is not moving). If a turbulence model is specified, the wall-normal distance boundary condition must be set to zero at the wall. The boundary conditions for the different turbulence variables depend on the model selected. For the Spalart-Allmaras model, the turbulent eddy viscosity, $\tilde{\nu}$, is set to zero at the wall. For the RNG k–ε model, the wall boundary conditions are automatically

implemented by the solver using the wall-function approach; no user settings for k or ε are required because they are prescribed automatically.

Slip wall

A slip wall is a surface where the fluid does not adhere to the wall but cannot penetrate it. This wall condition is modeled by specifying the wall-normal fluid velocity equal to the wall velocity (zero if the wall is not moving). This situation also represents a symmetry condition for fluid flow since the in-plane velocities can vary, but the out-of-plane velocity is zero. In cases where a moving boundary is being considered, an associated set of mesh displacement boundary conditions must be prescribed in conjunction with the surface fluid velocity to achieve the proper behavior. If a turbulence model is specified, the wall-normal distance boundary condition must be set to zero at the wall.

Infiltration wall

Infiltration at a surface permits the fluid to penetrate the surface while maintaining the no-slip condition. This wall condition is modeled by specifying the wall-normal velocity equal to the velocity representing the infiltration velocity, while the wall-tangent fluid velocity is equal to the wall velocity (zero if the wall is not moving). In the special case when a turbulence model is implemented, the wall-normal distance boundary condition must be set to zero at the wall. If the Spalart-Allmaras turbulence model is enabled, you can specify the value of the Spalart-Allmaras turbulent eddy viscosity, $\tilde{\nu}$, that is allowed at the wall due to infiltration. If the RNG k–ε model is implemented, you can prescribe values at the wall for the turbulent kinetic energy, k, and the dissipation rate, ε.

Prescribed temperature

Temperatures can be prescribed at a wall. By default, if no temperature is prescribed at a wall, a perfectly insulated boundary is specified automatically. For multiphysics applications such as conjugate heat transfer, a variable temperature condition is imposed automatically using a co-simulation region (for more information, see "Preparing an Abaqus analysis for co-simulation," Section 16.1.2).

Prescribed displacement

Abaqus/CFD provides the capability to perform both deforming-mesh and fluid-structure interaction (FSI) simulations using an arbitrary Lagrangian Eulerian (ALE) methodology for the fluid flow. For FSI and deforming-mesh problems, typically some portion of the fluid domain is deformed consistent with a boundary motion. To manage the mesh motion, you must prescribe displacement boundary conditions on the mesh. For FSI problems, displacement boundary conditions are not permitted at the co-simulation region because these conditions are prescribed automatically.

Input File Usage: *BOUNDARY
node or node set, *first degree of freedom*, *last degree of freedom*, *magnitude*

where *first degree of freedom* is 1 for the x-displacement, 2 for the y-displacement, or 3 for the z-displacement.

Abaqus/CAE Usage: Load module: **Create Boundary Condition**: **Step**: ***flow_step***: **Category**: **Mechanical**: **Displacement/Rotation**: select regions and toggle on the degree or degrees of freedom

Defining boundary conditions that vary with time

The prescribed magnitude of the boundary conditions can vary with time during a step according to an amplitude definition ("Amplitude curves," Section 32.1.2).

Input File Usage: Use both of the following options to define the prescribed displacement at a moving boundary:

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

Use both of the following options to define inflow and outflow boundary conditions and wall boundary conditions that vary with time:

```
*AMPLITUDE, NAME=name
*FLUID BOUNDARY, AMPLITUDE=name
```

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

Boundary condition propagation

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

Modifying boundary conditions

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

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

```
*BOUNDARY
*BOUNDARY, OP=MOD
*FLUID BOUNDARY
*FLUID BOUNDARY, OP=MOD
```

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

Removing boundary conditions

If you choose to remove any boundary condition in a step, no boundary conditions will be propagated from the previous general step. Therefore, all boundary conditions that are in effect during this step must be respecified.

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

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

∗BOUNDARY, OP=NEW

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

∗FLUID BOUNDARY, OP=NEW

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

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

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

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

32.4 Loads

- “Applying loads: overview,” Section 32.4.1
- “Concentrated loads,” Section 32.4.2
- “Distributed loads,” Section 32.4.3
- “Thermal loads,” Section 32.4.4
- “Electromagnetic loads,” Section 32.4.5
- “Acoustic and shock loads,” Section 32.4.6
- “Pore fluid flow,” Section 32.4.7

32.4.1 APPLYING LOADS: OVERVIEW

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CFD Abaqus/CAE

References

- "General and linear perturbation procedures," Section 6.1.2
- "Prescribed conditions: overview," Section 32.1.1
- "Concentrated loads," Section 32.4.2
- "Distributed loads," Section 32.4.3
- "Thermal loads," Section 32.4.4
- "Electromagnetic loads," Section 32.4.5
- "Acoustic and shock loads," Section 32.4.6
- "Pore fluid flow," Section 32.4.7
- "Creating and modifying prescribed conditions," Section 16.4 of the Abaqus/CAE User's Manual
- "Using the load editors," Section 16.9 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

External loading can be applied in the following forms:

- Concentrated or distributed tractions.
- Concentrated or distributed fluxes.
- Incident wave loads.

Many types of distributed loads are provided; they depend on the element type and are described in Part VI, "Elements." This section discusses general concepts that apply to all types of loading; see "Prescribed conditions: overview," Section 32.1.1, for general information that applies to all types of prescribed conditions.

Concentrated and distributed tractions are discussed in "Concentrated loads," Section 32.4.2, and "Distributed loads," Section 32.4.3, respectively. Thermal loading (heat flux) is discussed in "Thermal loads," Section 32.4.4. Electromagnetic loads are discussed in "Electromagnetic loads," Section 32.4.5. Loads due to incident wave fields such as due to sound sources or an underwater explosion are discussed in "Acoustic and shock loads," Section 32.4.6. Pore fluid flow is discussed in "Pore fluid flow," Section 32.4.7. All other load types, which are applicable to only a single type of analysis, are discussed in the appropriate sections in Part III, "Analysis Procedures, Solution, and Control."

In some situations, concentrated loads and some commonly used distributed loads (such as pressure applied on a surface) may rotate during a geometrically nonlinear analysis. Such loads are known as follower loads; further details on follower loads can be found in "Follower loads in large-displacement analysis;" "Specifying concentrated follower forces" in "Concentrated loads," Section 32.4.2; "Follower

surface loads" in "Distributed loads," Section 32.4.3; and "Follower edge and line loads" in "Distributed loads," Section 32.4.3. Follower loads may also lead to an unsymmetric contribution to the stiffness matrix, which is generally referred to as the load stiffness; some issues related to the load stiffness contribution are discussed in "Improving the rate of convergence in large-displacement implicit analysis" in "Concentrated loads," Section 32.4.2, and "Improving the rate of convergence in large-displacement implicit analysis" in "Distributed loads," Section 32.4.3.

Element-based versus surface-based distributed loads

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

Element-based loads

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

Surface-based loads

Use surface-based loads to prescribe a distributed load on a geometric surface or geometric edge. With surface-based loads you must specify the surface or edge name and the distributed load type. The surface or edge, which contains the element and face information, is defined as described in "Element-based surface definition," Section 2.3.2. In Abaqus/CAE surfaces can be defined as collections of geometric faces and edges or collections of element faces and edges. This method of prescribing a distributed load facilitates user input for complex models. It can be used with most element types for which a valid surface can be defined. You can specify in the surface definition how the distributed load is applied to the boundary of an adaptive mesh domain in Abaqus/Explicit (see "Defining ALE adaptive mesh domains in Abaqus/Explicit," Section 12.2.2).

Varying the magnitude of a load

The magnitude of a load is usually defined by the input data. The variation of the load magnitude during a step can be defined by the default amplitude variation for the step (see "Prescribed conditions: overview," Section 32.1.1); by a user-defined amplitude curve (see "Amplitude curves," Section 32.1.2); or, in some cases, by user subroutine **DLOAD**, **UDECURRENT**, **UDSECURRENT**, **UTRACLOAD**, or **VDLOAD**.

Loading during general analysis steps

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

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

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

Modifying loads

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

All loads defined in previous steps remain unchanged unless they are redefined. When a load is left unchanged, the following rules apply:

- If the associated amplitude was specified in terms of total time, the load continues to follow the amplitude definition.
- If no amplitude was associated with the load or if the amplitude was given in terms of step time, the load remains constant at the magnitude associated with the end of the previous step.

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

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

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

Removing loads

If you choose to remove any load of a particular type (concentrated load, element-based distributed load, surface-based distributed load, etc.) in a step, no loads of that type will be propagated from the previous general step. All loads of that type that are in effect during this step must be respecified. To redefine a load, the node, element, node set, element set, or surface name must be specified in exactly the same way and the load type must be identical. Refer to "Prescribed conditions: overview," Section 32.1.1, for a discussion of amplitude variations when removing loads.

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

**LOADING OPTION*, OP=NEW

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

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

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

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

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

Example

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

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

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

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

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

```
 1, 3, 5.
*DLOAD, OP=MOD
 A2, BX, 50.
*END STEP
```

Follower loads in large-displacement analysis

In large-displacement analysis distributed loads will be treated as follower forces when appropriate. For beam and shell elements point (concentrated) loads may be fixed in direction or they may rotate with the structure depending on whether you specify follower forces for the load (see "Concentrated loads," Section 32.4.2). Follower loads defined at a rigid body tie node rotate with the rigid body in Abaqus/Explicit.

Loading during linear perturbation steps

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

In consecutive linear perturbation steps, the perturbation of load that applies to each step must be defined completely within that step—the analysis within each such step always starts from the base state (except when you specify that a modal dynamic step should use the initial conditions from the immediately preceding step—see "Transient modal dynamic analysis," Section 6.3.7).

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

Loading during linear (mode-based) dynamics procedures

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

32.4.2 CONCENTRATED LOADS

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CFD Abaqus/CAE

References

- "Applying loads: overview," Section 32.4.1
- *CLOAD
- "Defining a concentrated force," Section 16.9.1 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a moment," Section 16.9.2 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a generalized plane strain load," Section 16.9.10 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a fluid reference pressure," Section 16.9.23 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Concentrated loads:

- apply concentrated forces and moments to nodal degrees of freedom; and
- can be fixed in direction; or
- can rotate as the node rotates (referred to as follower forces), resulting in an additional, and possibly unsymmetric, contribution to the load stiffness

In steady-state dynamic analysis both real and imaginary concentrated loads can be applied (see "Direct-solution steady-state dynamic analysis," Section 6.3.4, and "Mode-based steady-state dynamic analysis," Section 6.3.8, for details).

Multiple concentrated load cases can be defined in random response analysis (see "Random response analysis," Section 6.3.11, for details).

Concentrated loads are also used to apply the pressure-conjugate at nodes with pressure degree of freedom in acoustic analysis (see "Acoustic and shock loads," Section 32.4.6) and to specify a fluid reference pressure for incompressible flow (see "Incompressible fluid dynamic analysis," Section 6.6.2).

Actuation loads in connector elements can be defined as connector loads, applied similarly to concentrated loads. See "Connector actuation," Section 30.1.3, for more detailed information.

The procedures in which these loads can be used are outlined in "Prescribed conditions: overview," Section 32.1.1. See "Applying loads: overview," Section 32.4.1, for general information that applies to all types of loading.

Concentrated loads

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

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

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

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

Specifying concentrated follower forces

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

Follower loads lead to an unsymmetric contribution to the stiffness matrix that is generally referred to as the load stiffness. Some issues associated with the load stiffness contribution are discussed in "Improving the rate of convergence in large-displacement implicit analysis."

Input File Usage: *CLOAD, FOLLOWER

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

Defining the values of concentrated nodal force from a user-specified file

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

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

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

Specifying a fluid reference pressure

For incompressible fluid dynamic analyses in Abaqus/CFD, when no other pressure condition is prescribed, you must specify a fluid reference pressure at one node to set the hydrostatic pressure level. Multiple reference pressures can be specified, but only the last specified hydrostatic pressure load is applied. For more information, see "Incompressible fluid dynamic analysis," Section 6.6.2, and "Boundary conditions in Abaqus/CFD," Section 32.3.2.

Input File Usage: ∗CLOAD
node number or node set, HP, *magnitude*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Fluid** for the **Category** and **Fluid reference pressure** for the **Types for Selected Step**

Defining time-dependent concentrated loads

The prescribed magnitude of a concentrated load can vary with time during a step according to an amplitude definition, as described in "Prescribed conditions: overview," Section 32.1.1. If different variations are needed for different loads, each load can refer to its own amplitude.

Modifying concentrated loads

Concentrated loads can be added, modified, or removed as described in "Applying loads: overview," Section 32.4.1.

Improving the rate of convergence in large-displacement implicit analysis

When concentrated follower forces are specified in a geometrically nonlinear static and dynamic analysis, the unsymmetric matrix storage and solution scheme should normally be used. See "Procedures: overview," Section 6.1.1, for more information on the unsymmetric matrix storage and solution scheme.

32.4.3 DISTRIBUTED LOADS

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CFD Abaqus/CAE

References

- "Applying loads: overview," Section 32.4.1
- *DLOAD
- *DSLOAD
- "Defining a pressure load," Section 16.9.3 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a shell edge load," Section 16.9.4 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a surface traction load," Section 16.9.5 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a pipe pressure load," Section 16.9.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a body force," Section 16.9.7 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a line load," Section 16.9.8 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a gravity load," Section 16.9.9 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a rotational body force," Section 16.9.11 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Distributed loads:

- can be prescribed on element faces, element bodies, or element edges;
- can be prescribed over geometric surfaces or geometric edges;
- require that an appropriate distributed load type be specified—see Part VI, "Elements," for definitions of the distributed load types available for particular elements; and
- may be of follower type, which can rotate during a geometrically nonlinear analysis and result in an additional (often unsymmetric) contribution to the stiffness matrix that is generally referred to as the load stiffness.

The procedures in which these loads can be used are outlined in "Prescribed conditions: overview," Section 32.1.1. See "Applying loads: overview," Section 32.4.1, for general information that applies to all types of loading.

Follower loads are discussed further in "Follower surface loads" and "Follower edge and line loads." The contribution of follower loads to load stiffness is discussed in "Improving the rate of convergence in large-displacement implicit analysis."

In steady-state dynamic analysis both real and imaginary distributed loads can be applied (see "Direct-solution steady-state dynamic analysis," Section 6.3.4, and "Mode-based steady-state dynamic analysis," Section 6.3.8, for details).

Incident wave loading is used to apply distributed loads for the special case of loads associated with a wave traveling through an acoustic medium. Inertia relief is used to apply inertia-based loading in Abaqus/Standard. These load types are discussed in "Acoustic and shock loads," Section 32.4.6, and "Inertia relief," Section 11.1.1, respectively. Abaqus/Aqua load types are discussed in "Abaqus/Aqua analysis," Section 6.11.1.

Defining time-dependent distributed loads

The prescribed magnitude of a distributed load can vary with time during a step according to an amplitude definition, as described in "Prescribed conditions: overview," Section 32.1.1. If different variations are needed for different loads, each load can refer to its own amplitude definition.

Modifying distributed loads

Distributed loads can be added, modified, or removed as described in "Applying loads: overview," Section 32.4.1.

Improving the rate of convergence in large-displacement implicit analysis

In large-displacement analyses in Abaqus/Standard some distributed load types introduce unsymmetric load stiffness matrix terms. Examples are hydrostatic pressure, pressure applied to surfaces with free edges, Coriolis force, rotary acceleration force, and distributed edge loads and surface tractions modeled as follower loads. In such cases using the unsymmetric matrix storage and solution scheme for the analysis step may improve the convergence rate of the equilibrium iterations. See "Procedures: overview," Section 6.1.1, for more information on the unsymmetric matrix storage and solution scheme.

Defining distributed loads in a user subroutine

Nonuniform distributed loads such as a nonuniform body force in the X-direction can be defined by means of user subroutine **DLOAD** in Abaqus/Standard or **VDLOAD** in Abaqus/Explicit. When an amplitude reference is used with a nonuniform load defined in user subroutine **VDLOAD**, the current value of the amplitude function is passed to the user subroutine at each time increment in the analysis. **DLOAD** and **VDLOAD** are not available for surface tractions, edge tractions, or edge moments.

In Abaqus/Standard nonuniform distributed surface tractions, edge tractions, and edge moments can be defined by means of user subroutine **UTRACLOAD**. User subroutine **UTRACLOAD** allows you to define a nonuniform magnitude for surface tractions, edge tractions, and edge moments, as well as nonuniform loading directions for general surface tractions, shear tractions, and general edge tractions.

Nonuniform distributed surface tractions, edge tractions, and edge moments are not currently supported in Abaqus/Explicit.

When the user subroutine is used, the external work is calculated based only on the current magnitude of the distributed load since the incremental value for the distributed load is not defined.

Specifying the region to which a distributed load is applied

As discussed in "Applying loads: overview," Section 32.4.1, distributed loads can be defined as element-based or surface-based. Element-based distributed loads can be prescribed on element bodies, element surfaces, or element edges. Surface-based distributed loads can be prescribed directly on geometric surfaces or geometric edges.

Three types of distributed loads can be defined: body loads, surface loads, and edge loads. Distributed body loads are always element-based. Distributed surface loads and distributed edge loads can be element-based or surface-based. Table 32.4.3–1 summarizes the regions on which each load type can be prescribed. In Abaqus/CAE distributed loads are specified by selecting the region in the viewport or from a list of surfaces. In the Abaqus input file different options are used depending on the type of region to which the load is applied, as illustrated in the following sections.

Table 32.4.3–1 Regions on which the different load types can be prescribed.

Load type	Load definition	Input file region	Abaqus/CAE region
Body loads	Element-based	Element bodies	Volumetric bodies
Surface loads	Element-based	Element surfaces	Surfaces defined as collections of geometric faces or element faces (excluding analytical rigid surfaces)
	Surface-based	Geometric element-based surfaces	
Edge loads (including beam line loads)	Element-based	Element edges	Surfaces defined as collections of geometric edges or element edges
	Surface-based	Geometric edge-based surfaces	

Body forces

Body loads, such as gravity, centrifugal, Coriolis, and rotary acceleration loads, are applied as element-based loads. The units of a body force are force per unit volume.

Table 32.4.3–2 lists all of the distributed body load types that are available in Abaqus, along with the corresponding load type labels.

Table 32.4.3–2 Distributed body load types.

Load description	Load type label for element-based loads	Load type label for surface-based loads	Abaqus/CAE load type
Body force in global X-, Y-, and Z-directions	BX, BY, BZ	N/A	**Body force**

Load description	Load type label for element-based loads	Load type label for surface-based loads	Abaqus/CAE load type
Nonuniform body force in global *X*-, *Y*-, and *Z*-directions	BXNU, BYNU, BZNU	N/A	**Body force**
Body force in radial and axial directions (only for axisymmetric elements)	BR, BZ	N/A	
Nonuniform body force in radial and axial directions (only for axisymmetric elements)	BRNU, BZNU	N/A	
Viscous body force in global *X*-, *Y*-, and *Z*-directions (available only in Abaqus/Explicit)	VBF	N/A	Not supported
Stagnation body force in global *X*-, *Y*-, and *Z*-directions (available only in Abaqus/Explicit)	SBF	N/A	
Gravity loading	GRAV	N/A	**Gravity**
Centrifugal load (magnitude is input as $\rho\omega^2$, where ρ is the mass density per unit volume and ω is the angular velocity)	CENT	N/A	Not supported
Centrifugal load (magnitude is input as ω^2, where ω is the angular velocity)	CENTRIF	N/A	**Rotational body force**
Coriolis force	CORIO	N/A	**Coriolis force**
Rotary acceleration load	ROTA	N/A	**Rotational body force**

Specifying general body forces

You can specify body forces on any elements in the global *X*-, *Y*-, or *Z*-direction. You can specify body forces on axisymmetric elements in the radial or axial direction.

Input File Usage: Use the following option to define a body force in the global *X*-, *Y*-, or *Z*-direction:

∗DLOAD
element number or element set, *load type label*, *magnitude*

where *load type label* is BX, BY, BZ, BXNU, BYNU, or BZNU.

Use the following option to define a body force in the radial or axial direction on axisymmetric elements:

∗DLOAD
element number or element set, *load type label*, *magnitude*

where *load type label* is BR, BZ, BRNU, or BZNU.

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Body force** for the **Types for Selected Step**

Specifying viscous body force loads in Abaqus/Explicit

Viscous body force loads are defined by

$$\mathbf{f_v} = -\mathbf{c_{vb}}\,(\mathbf{v} - \mathbf{v_{ref}})\mathbf{V_e},$$

where $\mathbf{f_v}$ is the viscous force applied to the body; c_{vb} is the viscosity, given as the magnitude of the load; $\mathbf{v}$ is the velocity of the point on the body where the force is being applied; $\mathbf{v}_{ref}$ is the velocity of the reference node; and V_e is the element volume.

Viscous body force loading can be thought of as mass-proportional damping in the sense that it gives a damping contribution proportional to the mass for an element if the coefficient c_{vb} is chosen to be a small value multiplied by the material density ρ (see "Material damping," Section 25.1.1). Viscous body force loading provides an alternative way to define mass-proportional damping as a function of relative velocities and a step-dependent damping coefficient.

Input File Usage: Use the following option to define a viscous body force load:

∗DLOAD, REF NODE=*reference_node*
element number or element set, VBF, *magnitude*

Abaqus/CAE Usage: Viscous body force loads are not supported in Abaqus/CAE.

Specifying stagnation body force loads in Abaqus/Explicit

Stagnation body force loads are defined by

$$\mathbf{f_s} = -\mathbf{c_{sb}}\,(\mathbf{v} - \mathbf{v_{ref}})^{\mathbf{2}}\mathbf{V_e},$$

where $\mathbf{f_s}$ is the stagnation body force applied to the body; c_{sb} is the factor, given as the magnitude of the load; $\mathbf{v}$ is the velocity of the point on the body where the body force is being applied; $\mathbf{v}_{ref}$ is the velocity of the reference node; and V_e is the element volume. The coefficient c_{sb} should be very small to avoid excessive damping and a dramatic drop in the stable time increment.

Input File Usage: Use the following option to define a stagnation body force load:

∗DLOAD, REF NODE=*reference_node*
element number or element set, SBF, *magnitude*

Abaqus/CAE Usage: Stagnation body force loads are not supported in Abaqus/CAE.

Specifying gravity loading

Gravity loading (uniform acceleration in a fixed direction) is specified by using the gravity distributed load type and giving the gravity constant as the magnitude of the load. The direction of the gravity field is specified by giving the components of the gravity vector in the distributed load definition. Abaqus uses the user-specified material density (see "Density," Section 20.2.1), together with the magnitude and direction, to calculate the loading. The magnitude of the gravity load can vary with time during a step according to an amplitude definition, as described in "Prescribed conditions: overview," Section 32.1.1. However, the direction of the gravity field is always applied at the beginning of the step and remains fixed during the step.

You need not specify an element or an element set as is customary for the specification of other distributed loads. Abaqus/Standard and Abaqus/Explicit automatically collect all elements in the model that have mass contributions (including point mass elements but excluding rigid elements) in an element set called **`_Whole_Model_Gravity_Elset`** and apply the gravity loads to the elements in this element set. Abaqus/CFD applies the gravity loading to all user-defined elements.

In Abaqus/CFD gravity loading defines the gravity vector used with a Boussinesq-type body force in buoyancy driven flow. You must activate the energy equation for incompressible flow and define thermal expansion to specify the thermal expansion coefficient (see "Incompressible fluid dynamic analysis," Section 6.6.2, and "Thermal expansion," Section 25.1.2). Gravity loading can be used only in conjunction with the energy equation and will be ignored if used without the energy equation; general body forces can be defined for incompressible flow without the energy equation.

When gravity loading is used with substructures, the density must be defined and unit gravity load vectors must be calculated when the substructure is created (see "Defining substructures," Section 10.1.2).

Input File Usage: Use the following option to define a gravity load:

∗DLOAD
element number or element set, GRAV, *gravity constant*, *comp1, comp2, comp3*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Gravity** for the **Types for Selected Step**

Specifying loads due to rotation of the model in Abaqus/Standard

Centrifugal loads, Coriolis forces, and rotary acceleration loads can be applied in Abaqus/Standard by specifying the appropriate distributed load type in an element-based distributed load definition. These loading options are primarily intended for replicating dynamic loads while performing analyses other than implicit dynamics using direct integration ("Dynamic stress/displacement analysis," Section 6.3). In an implicit dynamic procedure inertia loads due to rotations come about naturally due to equilibrium. Applying distributed centrifugal, Coriolis, and rotary inertia loads in an implicit dynamic analysis may lead to non-physical loads and should be used carefully.

Centrifugal loads

Centrifugal load magnitudes can be specified as ω^2, where ω is the angular velocity in radians per time. Abaqus/Standard uses the specified material density (see "Density," Section 20.2.1), together with the load magnitude and the axis of rotation, to calculate the loading. Alternatively, a centrifugal load magnitude can be given as $\rho\omega^2$, where ρ is the material density (mass per unit volume) for solid or shell elements or the mass per unit length for beam elements and ω is the angular velocity in radians per time. This type of centrifugal load formulation does not account for large volume changes. The two centrifugal load types will produce slightly different local results for first-order elements; $\rho\omega^2$ uses a consistent mass matrix, and ω^2 uses a lumped mass matrix in calculating the load forces and load stiffnesses.

The magnitude of the centrifugal load can vary with time during a step according to an amplitude definition, as described in "Prescribed conditions: overview," Section 32.1.1. However, the position and orientation of the axis around which the structure rotates, which is defined by giving a point on the axis and the axis direction, are always applied at the beginning of the step and remain fixed during the step.

Input File Usage: Use either of the following options to define a centrifugal load:

*DLOAD
element number or element set, CENTRIF, ω^2, *coord1, coord2, coord3, comp1, comp2, comp3*
*DLOAD
element number or element set, CENT, $\rho\omega^2$, *coord1, coord2, coord3, comp1, comp2, comp3*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Rotational body force** for the **Types for Selected Step**: **Load effect: Centrifugal**

Coriolis forces

Coriolis force is defined by specifying the Coriolis distributed load type and giving the load magnitude as $\rho\omega$, where ρ is the material density (mass per unit volume) for solid and shell elements or the mass per unit length for beam elements and ω is the angular velocity in radians per time. The magnitude of the Coriolis load can vary with time during a step according to an amplitude definition, as described in "Prescribed conditions: overview," Section 32.1.1. However, the position and orientation of the axis

around which the structure rotates, which is defined by giving a point on the axis and the axis direction, are always applied at the beginning of the step and remain fixed during the step.

In a static analysis Abaqus computes the translational velocity term in the Coriolis loading by dividing the incremental displacement by the current time increment.

The Coriolis load formulation does not account for large volume changes.

Input File Usage: Use the following option to define a Coriolis load:

∗DLOAD
element number or element set, CORIO, $\rho\omega$, *coord1, coord2, coord3, comp1, comp2, comp3*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Coriolis force** for the **Types for Selected Step**

Rotary acceleration loads

Rotary acceleration loads are defined by specifying the rotary acceleration distributed load type and giving the rotary acceleration magnitude, α, in radians/time^2, which includes any precessional motion effects. The axis of rotary acceleration must be defined by giving a point on the axis and the axis direction. Abaqus/Standard uses the specified material density (see "Density," Section 20.2.1), together with the rotary acceleration magnitude and axis of rotary acceleration, to calculate the loading. The magnitude of the load can vary with time during a step according to an amplitude definition, as described in "Prescribed conditions: overview," Section 32.1.1. However, the position and orientation of the axis around which the structure rotates are always applied at the beginning of the step and remain fixed during the step.

Rotary acceleration loads are not applicable to axisymmetric elements.

Input File Usage: Use the following option to define a rotary acceleration load:

∗DLOAD
element number or element set, ROTA, α, *coord1, coord2, coord3, comp1, comp2, comp3*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Rotational body force** for the **Types for Selected Step**: **Load effect: Rotary acceleration**

Specifying general rigid-body acceleration loading in Abaqus/Standard

General rigid-body acceleration loading can be specified in Abaqus/Standard by using a combination of the gravity, centrifugal (ω^2), and rotary acceleration load types.

Surface tractions and pressure loads

General or shear surface tractions and pressure loads can be applied in Abaqus as element-based or surface-based distributed loads. The units of these loads are force per unit area.

Table 32.4.3–3 lists all of the distributed surface load types that are available in Abaqus, along with the corresponding load type labels. Part VI, "Elements," lists the distributed surface load types that are available for particular elements and the Abaqus/CAE load support for each load type. For some

element-based loads you must identify the face of the element upon which the load is prescribed in the load type label (for example, P*n* or P*n*NU for continuum elements).

Follower surface loads

By definition, the line of action of a *follower* surface load rotates with the surface in a geometrically nonlinear analysis. This is in contrast to a *non-follower* load, which always acts in a fixed global direction.

With the exception of general surface tractions, all the distributed surface loads listed in Table 32.4.3–3 are modeled as follower loads. The hydrostatic and viscous pressures listed in Table 32.4.3–3 always act normal to the surface in the current configuration, the shear tractions always act tangent to the surface in the current configuration, and the internal and external pipe pressures follow the motion of the pipe elements.

General surface tractions can be specified to be follower or non-follower loads. There is no difference between a follower and a non-follower load in a geometrically linear analysis since the configuration of the body remains fixed. The difference between a follower and non-follower general surface traction is illustrated in the next section through an example.

Input File Usage: Use one of the following options to define general surface tractions as follower loads (the default):

*DLOAD, FOLLOWER=YES
*DSLOAD, FOLLOWER=YES

Use one of the following options to define general surface tractions as non-follower loads:

*DLOAD, FOLLOWER=NO
*DSLOAD, FOLLOWER=NO

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction: General**, toggle on or off **Follow rotation**

Specifying general surface tractions

General surface tractions allow you to specify a surface traction, $\mathbf{t}$, acting on a surface S. The resultant load, $\mathbf{f}$, is computed by integrating $\mathbf{t}$ over S:

$$\mathbf{f} = \int_S \mathbf{t}\, dS = \int_S \alpha\, \hat{\mathbf{t}}\, dS,$$

where α is the magnitude and $\hat{\mathbf{t}}$ is the direction of the load. To define a general surface traction, you must specify both a load magnitude, α, and the direction of the load with respect to the reference configuration, $\mathbf{t}_{user}$. The magnitude and direction can also be specified in user subroutine **UTRACLOAD**. The specified traction directions are normalized by Abaqus and, thus, do not contribute to the magnitude of the load:

$$\hat{\mathbf{t}}^o = \frac{\mathbf{t}_{user}}{\|\mathbf{t}_{user}\|}.$$

Table 32.4.3–3 Distributed surface load types.

Load description	Load type label for element-based loads	Load type label for surface-based loads	Abaqus/CAE load type
General surface traction	TRVEC*n*, TRVEC	TRVEC	**Surface traction**
Shear surface traction	TRSHR*n*, TRSHR	TRSHR	
Nonuniform general surface traction	TRVEC*n*NU, TRVECNU	TRVECNU	**Surface traction** (surface-based loads only)
Nonuniform shear surface traction	TRSHR*n*NU, TRSHRNU	TRSHRNU	
Pressure	P*n*, P	P	**Pressure**
Nonuniform pressure	P*n*NU, PNU	PNU	**Pressure** (surface-based loads only)
Hydrostatic pressure (available only in Abaqus/Standard)	HP*n*, HP	HP	
Viscous pressure (available only in Abaqus/Explicit)	VP*n*, VP	VP	
Stagnation pressure (available only in Abaqus/Explicit)	SP*n*, SP	SP	
Hydrostatic internal and external pressure (only for PIPE and ELBOW elements)	HPI, HPE	N/A	**Pipe pressure**
Uniform internal and external pressure (only for PIPE and ELBOW elements)	PI, PE	N/A	
Nonuniform internal and external pressure (only for PIPE and ELBOW elements)	PINU, PENU	N/A	

Input File Usage: Use one of the following options to define a general surface traction:

∗DLOAD
element number or element set, *load type label*, *magnitude*, *direction components*

where *load type label* is TRVEC*n*, TRVEC, TRVEC*n*NU, or TRVECNU.

∗DSLOAD
surface name, TRVEC or TRVECNU, *magnitude*, *direction components*

Abaqus/CAE Usage: Use the following input to define an element-based general surface traction:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction: General**, **Distribution**: select an analytical field

Use the following input to define a surface-based general surface traction:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction: General**, **Distribution**: **Uniform** or **User-defined**

Nonuniform element-based general surface traction is not supported in Abaqus/CAE.

Defining the direction vector with respect to a local coordinate system

By default, the components of the traction vector are specified with respect to the global directions. You can also refer to a local coordinate system (see "Orientations," Section 2.2.5) for the direction components of these tractions. See "Examples: using a local coordinate system to define shear directions" below for an example of a traction load defined with respect to a local coordinate system.

Input File Usage: Use one of the following options to specify a local coordinate system:

*DLOAD, ORIENTATION=*name*
*DSLOAD, ORIENTATION=*name*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: select **CSYS: Picked** and click **Edit** to pick a local coordinate system, or select **CSYS: User-defined** to enter the name of a user subroutine that defines a local coordinate system

Rotation of the traction vector direction

The traction load acts in the fixed direction $\hat{\mathbf{t}} = \hat{\mathbf{t}}^o$ in a geometrically linear analysis or if a non-follower load is specified in a geometrically nonlinear analysis (which includes a perturbation step about a geometrically nonlinear base state).

If a follower load is specified in a geometrically nonlinear analysis, the traction load rotates rigidly with the surface using the following algorithm. The reference configuration traction vector, $\mathbf{t}^o = \alpha\,\hat{\mathbf{t}}^o$, is decomposed by Abaqus into two components: a normal component,

$$\alpha\,\hat{\mathbf{t}}^o \cdot \mathbf{N}\,\mathbf{N} \;=\; \alpha_n\,\mathbf{N}\,,$$

and a tangential component,

$$\alpha\,(\hat{\mathbf{t}}^o - \hat{\mathbf{t}}^o \cdot \mathbf{N}\,\mathbf{N}) \;=\; \alpha_d\,\mathbf{D}\,,$$

where $\mathbf{N}$ is the unit reference surface normal and $\mathbf{D}$ is the unit projection of $\hat{\mathbf{t}}^o$ onto the reference surface. The applied traction in the current configuration is then computed as

$$\mathbf{t} = \alpha_n \mathbf{n} + \alpha_d \mathbf{d},$$

where $\mathbf{n}$ is the normal to the surface in the current configuration and $\mathbf{d}$ is the image of $\mathbf{D}$ rotated onto the current surface; i.e., $\mathbf{d} = \mathbf{R}\mathbf{D}$, where $\mathbf{R}$ is the standard rotation tensor obtained from the polar decomposition of the local two-dimensional surface deformation gradient $\mathbf{F} = \mathbf{R}\mathbf{U}$.

Examples: follower and non-follower tractions

The following two examples illustrate the difference between applying follower and non-follower tractions in a geometrically nonlinear analysis. Both examples refer to a single 4-node plane strain element (element 1). In Step 1 of the first example a follower traction load is applied to face 1 of element 1, and a non-follower traction load is applied to face 2 of element 1. The element is rotated rigidly 90° counterclockwise in Step 1 and then another 90° in Step 2. As illustrated in Figure 32.4.3–1, the follower traction rotates with face 1, while the non-follower traction on face 2 always acts in the global x-direction.

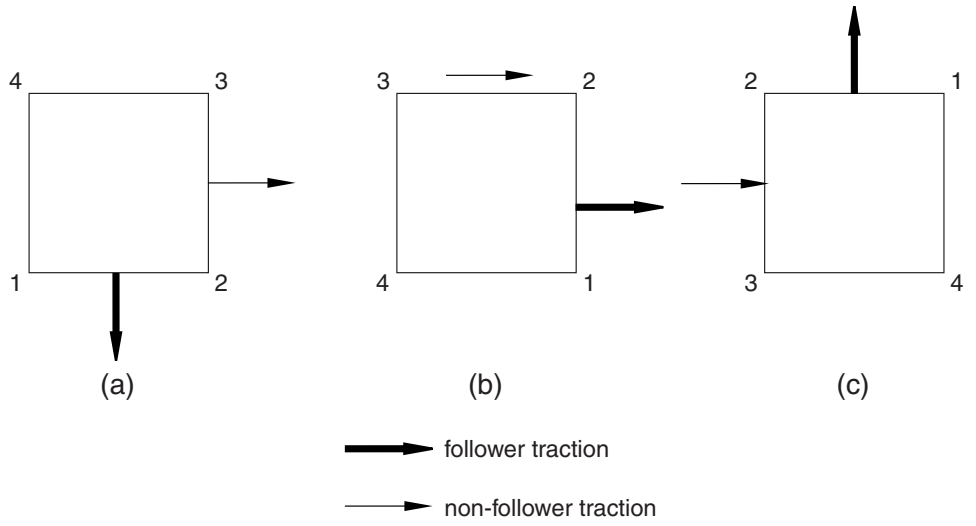


Figure 32.4.3–1 Follower and non-follower traction loads in a geometrically nonlinear analysis, load applied in Step 1: (a) beginning of Step 1; (b) end of Step 1, beginning of Step 2; (c) end of Step 2.

```
*STEP, NLGEOM
 Step 1 - Rotate square 90 degrees
...
*DLOAD, FOLLOWER=YES
 1, TRVEC1, 1., 0., -1., 0.
*DLOAD, FOLLOWER=NO
```

```
 1, TRVEC2, 1., 1., 0., 0.
*END STEP
*STEP, NLGEOM
 Step 2 - Rotate square another 90 degrees
...
*END STEP
```

In the second example the element is rotated 90° counterclockwise with no load applied in Step 1. In Step 2 a follower traction load is applied to face 1, and a non-follower traction load is applied to face 2. The element is then rotated rigidly by another 90°. The direction of the follower load is specified with respect to the original configuration. As illustrated in Figure 32.4.3–2, the follower traction rotates with face 1, while the non-follower traction on face 2 always acts in the global x-direction.

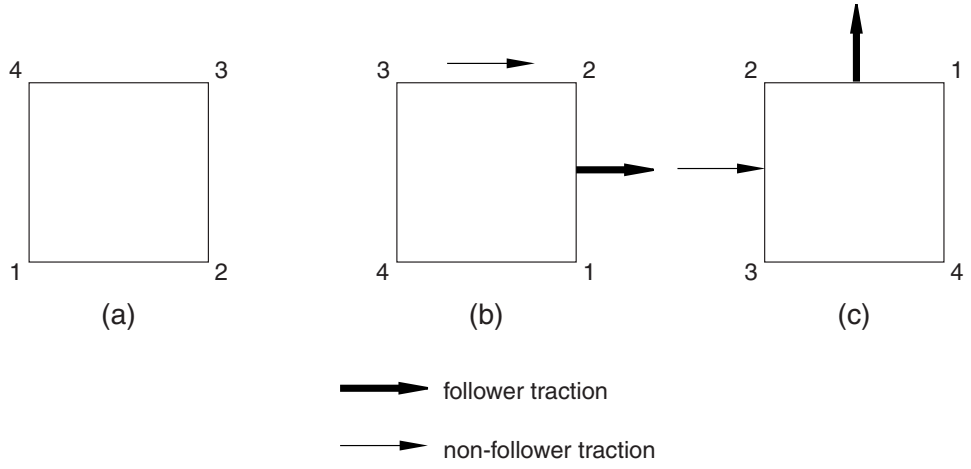


Figure 32.4.3–2 Follower and non-follower traction loads in a geometrically nonlinear analysis, load applied in Step 2: (a) beginning of Step 1; (b) end of Step 1, beginning of Step 2; (c) end of Step 2.

```
*STEP, NLGEOM
 Step 1 - Rotate square 90 degrees
...
*END STEP
*STEP, NLGEOM
 Step 2 - Rotate square another 90 degrees
*DLOAD, FOLLOWER=YES
 1, TRVEC1, 1., 0., -1., 0.
*DLOAD, FOLLOWER=NO
 1, TRVEC2, 1., 1., 0., 0.
```

```
...
*END STEP
```

Specifying shear surface tractions

Shear surface tractions allow you to specify a surface force per unit area, $\mathbf{t}_s$, that acts tangent to a surface S. The resultant load, $\mathbf{f}$, is computed by integrating $\mathbf{t}_s$ over S:

$$\mathbf{f} = \int_S \mathbf{t}_s \, dS = \int_S \alpha \, \mathbf{d} \, dS,$$

where α is the magnitude and $\mathbf{d}$ is a unit vector along the direction of the load. To define a shear surface traction, you must provide both the magnitude, α, and a direction, $\mathbf{t}_{user}$, for the load. The magnitude and direction vector can also be specified in user subroutine **UTRACLOAD**.

Abaqus modifies the traction direction by first projecting the user-specified vector, $\mathbf{t}_{user}$, onto the surface in the *reference* configuration,

$$\mathbf{t}_{user}^{po} = \mathbf{t}_{user} - \mathbf{t}_{user} \cdot \mathbf{NN},$$

where $\mathbf{N}$ is the reference surface normal. The specified traction is applied along the computed traction direction $\mathbf{D}$ tangential to the surface:

$$\mathbf{D} = \frac{\mathbf{t}_{user}^{po}}{||\mathbf{t}_{user}^{po}||}.$$

Consequently, a shear traction load is not applied at any point where $\mathbf{t}_{user}$ is normal to the reference surface.

The shear traction load acts in the fixed direction $\mathbf{d} = \mathbf{D}$ in a geometrically linear analysis. In a geometrically nonlinear analysis (which includes a perturbation step about a geometrically nonlinear base state), the shear traction vector will rotate rigidly; i.e., $\mathbf{d} = \mathbf{RD}$, where $\mathbf{R}$ is the standard rotation tensor obtained from the polar decomposition of the local two-dimensional surface deformation gradient $\mathbf{F} = \mathbf{RU}$.

Input File Usage: Use one of the following options to define a shear surface traction:

∗DLOAD
element number or element set, *load type label*, *magnitude*, *direction components*

where *load type label* is TRSHR*n*, TRSHR, TRSHR*n*NU, or TRSHRNU.

∗DSLOAD
surface name, TRSHR or TRSHRNU, *magnitude*, *direction components*

Abaqus/CAE Usage: Use the following input to define an element-based shear surface traction:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction: Shear**, **Distribution**: select an analytical field

Use the following input to define a surface-based general surface traction:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction: Shear**, **Distribution**: **Uniform** or **User-defined**

Nonuniform element-based shear surface traction is not supported in Abaqus/CAE.

Defining the direction vector with respect to a local coordinate system

By default, the components of the shear traction vector are specified with respect to the global directions. You can also refer to a local coordinate system (see "Orientations," Section 2.2.5) for the direction components of these tractions.

Input File Usage: Use one of the following options to specify a local coordinate system:

∗DLOAD, ORIENTATION=*name*
∗DSLOAD, ORIENTATION=*name*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: select **CSYS: Picked** and click **Edit** to pick a local coordinate system, or select **CSYS: User-defined** to enter the name of a user subroutine that defines a local coordinate system

Examples: using a local coordinate system to define shear directions

It is sometimes convenient to give shear and general traction directions with respect to a local coordinate system. The following two examples illustrate the specification of the direction of a shear traction on a cylinder using global coordinates in one case and a local cylindrical coordinate system in the other case. The axis of symmetry of the cylinder coincides with the global z-axis. A surface named **SURFA** has been defined on the outside of the cylinder.

In the first example the direction of the shear traction, $\mathbf{t}_{user} = (0., 1., 0.)$, is given in global coordinates. The sense of the resulting shear tractions using global coordinates is shown in Figure 32.4.3–3(a).

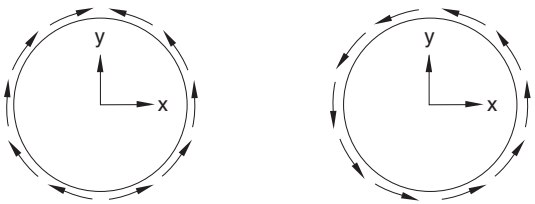


Figure 32.4.3–3 Shear tractions specified using global coordinates (a) and a local cylindrical coordinate system (b).

```
*STEP
 Step 1 - Specify shear directions in global coordinates
...
*DSLOAD
 SURFA, TRSHR, 1., 0., 1., 0.
...
*END STEP
```

In the second example the direction of the shear traction, $\mathbf{t}_{user} = (0., 1., 0.)$, is given with respect to a local cylindrical coordinate system whose axis coincides with the axis of the cylinder. The sense of the resulting shear tractions using the local cylindrical coordinate system is shown in Figure 32.4.3–3(b).

```
*ORIENTATION, NAME=CYLIN, SYSTEM=CYLINDRICAL
 0., 0., 0., 0., 0., 1.
...
*STEP
 Step 1 - Specify shear directions in local cylindrical coordinates
...
*DSLOAD, ORIENTATION=CYLIN
 SURFA, TRSHR, 1., 0., 1., 0.
...
*END STEP
```

Resultant loads due to surface tractions

You can choose to integrate surface tractions over the current or the reference configuration by specifying whether or not a constant resultant should be maintained.

In general, the constant resultant method is best suited for cases where the magnitude of the resultant load should not vary with changes in the surface area. However, it is up to you to decide which approach is best for your analysis. An example of an analysis using a constant resultant can be found in "Distributed traction and edge loads," Section 1.4.18 of the Abaqus Verification Manual.

Choosing not to have a constant resultant

If you choose not to have a constant resultant, the traction vector is integrated over the surface in the current configuration, a surface that in general deforms in a geometrically nonlinear analysis. By default, all surface tractions are integrated over the surface in the current configuration.

Input File Usage: Use one of the following options:

*DLOAD, CONSTANT RESULTANT=NO
*DSLOAD, CONSTANT RESULTANT=NO

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction is defined per unit deformed area**

Maintaining a constant resultant

If you choose to have a constant resultant, the traction vector is integrated over the surface in the reference configuration and then held constant.

Input File Usage: Use one of the following options:

*DLOAD, CONSTANT RESULTANT=YES
*DSLOAD, CONSTANT RESULTANT=YES

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Surface traction** for the **Types for Selected Step**: **Traction is defined per unit undeformed area**

Example

The constant resultant method has certain advantages when a traction is used to model a distributed load with a known constant resultant. Consider the case of modeling a uniform dead load, magnitude p, acting on a flat plate whose normal is in the $\mathbf{e}_2$-direction in a geometrically nonlinear analysis (Figure 32.4.3–4).

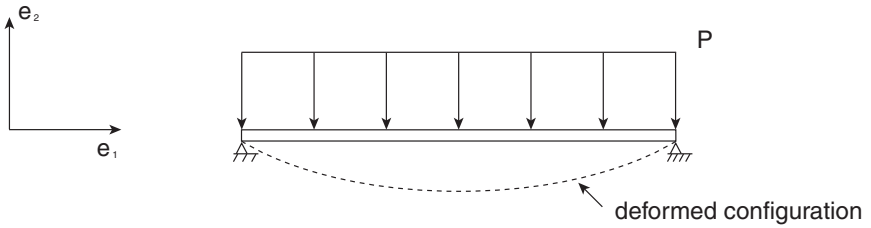


Figure 32.4.3–4 Dead load on a flat plate.

Such a model might be used to simulate a snow load on a flat roof. The snow load could be modeled as a distributed dead traction load $\mathbf{t} = -p\mathbf{e}_2$. Let S_o and S denote the total surface area of the plate in the reference and current configurations, respectively. With no constant resultant, the total integrated load on the plate, $\mathbf{f}$, is

$$\mathbf{f} = \int_S \mathbf{t}\, dS = \int_S -p\,\mathbf{e}_2\, dS = -p\,\mathbf{e}_2\, S.$$

In this case a uniform traction leads to a resultant load that increases as the surface area of the plate increases, which is not consistent with a fixed snow load. With the constant resultant method, the total integrated load on the plate is

$$\mathbf{f} = \int_{S_o} \mathbf{t}\, dS_o = \int_{S_o} -p\,\mathbf{e}_2\, dS_o = -p\,\mathbf{e}_2\, S_o.$$

In this case a uniform traction leads to a resultant that is equal to the pressure times the surface area in the reference configuration, which is more consistent with the problem at hand.

Specifying pressure loads

Distributed pressure loads can be specified on any two-dimensional, three-dimensional, or axisymmetric elements. Hydrostatic pressure loads can be specified in Abaqus/Standard on two-dimensional, three-dimensional, and axisymmetric elements. Viscous and stagnation pressure loads can be specified in Abaqus/Explicit on any elements.

Distributed pressure loads

Distributed pressure loads can be specified on any elements. For beam elements, a positive applied pressure results in a force vector acting along the particular local direction of the section or a global direction, whichever is specified. For conventional shell elements, the force vector points along the element SPOS normal. For continuum solid or a continuum shell elements with the distributed load on an explicitly identified facet, the force vector acts against the outward normal of that facet. Distributed pressure loads are not supported for pipe and elbow elements.

Distributed pressure loads can be specified on a surface formed over elements; a positive applied pressure results in a force vector acting against the local surface normal.

Input File Usage: Use one of the following options to define a pressure load:

*DLOAD
element number or element set, *load type label*, *magnitude*

where *load type label* is P*n*, P, P*n*NU, or PNU.

*DSLOAD
surface name, P or PNU, *magnitude*

Abaqus/CAE Usage: Use the following input to define an element-based pressure load:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pressure** for the **Types for Selected Step**: **Distribution**: select an analytical field or a discrete field

Use the following input to define a surface-based pressure load:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pressure** for the **Types for Selected Step**: **Uniform** or **User-defined**

Nonuniform element-based pressure loads are not supported in Abaqus/CAE.

Hydrostatic pressure loads on two-dimensional, three-dimensional, and axisymmetric elements in Abaqus/Standard

To define hydrostatic pressure in Abaqus/Standard, give the Z-coordinates of the zero pressure level (point a in Figure 32.4.3–5) and the level at which the hydrostatic pressure is defined (point b in Figure 32.4.3–5) in an element-based or surface-based distributed load definition. For levels above the zero pressure level, the hydrostatic pressure is zero.

In planar elements the hydrostatic head is in the Y-direction; for axisymmetric elements the Z-direction is the second coordinate.

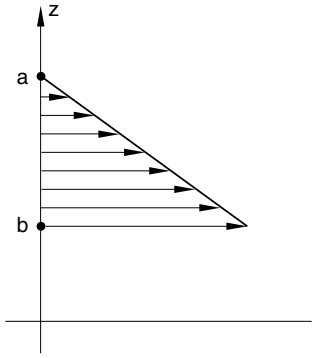


Figure 32.4.3–5 Hydrostatic pressure distribution.

Input File Usage: Use one of the following options to define a hydrostatic pressure load:

∗DLOAD
element number or element set, HP*n* or HP, *magnitude*, *Z-coordinate of point a*, *Z-coordinate of point b*

∗DSLOAD
surface name, HP, *magnitude*, *Z-coordinate of point a*, *Z-coordinate of point b*

Abaqus/CAE Usage: Use the following input to define a surface-based hydrostatic pressure load:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pressure** for the **Types for Selected Step**: **Distribution: Hydrostatic**

Element-based hydrostatic pressure loads are not supported in Abaqus/CAE.

Viscous pressure loads in Abaqus/Explicit

Viscous pressure loads are defined by

$$p = -c_v \left(\mathbf{v} - \mathbf{v}_{ref}\right) \cdot \mathbf{n},$$

where p is the pressure applied to the body; c_v is the viscosity, given as the magnitude of the load; $\mathbf{v}$ is the velocity of the point on the surface where the pressure is being applied; $\mathbf{v}_{ref}$ is the velocity of the reference node; and $\mathbf{n}$ is the unit outward normal to the element at the same point.

Viscous pressure loading is most commonly applied in structural problems when you want to damp out dynamic effects and, thus, reach static equilibrium in a minimal number of increments. A common example is the determination of springback in a sheet metal product after forming, in which case a viscous pressure would be applied to the faces of shell elements defining the sheet metal. An appropriate choice for the value of c_v is important for using this technique effectively.

To compute c_v, consider the infinite continuum elements described in "Infinite elements," Section 27.3.1. In explicit dynamics those elements achieve an infinite boundary condition by applying a viscous normal pressure where the coefficient c_v is given by ρc_d; ρ is the density of the material at the surface, and c_d is the value of the dilatational wave speed in the material (the infinite continuum elements also apply a viscous shear traction). For an isotropic, linear elastic material

$$c_d = \sqrt{\frac{\lambda + 2\mu}{\rho}} = \sqrt{\frac{E(1-\nu)}{\rho(1+\nu)(1-2\nu)}},$$

where λ and μ are Lamé's constants, $\boldsymbol{E}$ is Young's modulus, and ν is Poisson's ratio. This choice of the viscous pressure coefficient represents a level of damping in which pressure waves crossing the free surface are absorbed with no reflection of energy back into the interior of the finite element mesh.

For typical structural problems it is not desirable to absorb all of the energy (as is the case in the infinite elements). Typically c_v is set equal to a small percentage (perhaps 1 or 2 percent) of ρc_d as an effective way of minimizing ongoing dynamic effects. The c_v coefficient should have a positive value.

Input File Usage: Use one of the following options to define a viscous pressure load:

∗DLOAD, REF NODE=*reference_node*
element number or element set, VP*n* or VP, *magnitude*
∗DSLOAD, REF NODE=*reference_node*
surface name, VP, *magnitude*

Abaqus/CAE Usage: Use the following input to define a surface-based viscous pressure load:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pressure** for the **Types for Selected Step**: **Distribution: Viscous**, toggle on or off **Determine velocity from reference point**

Element-based viscous pressure loads are not supported in Abaqus/CAE.

Stagnation pressure loads in Abaqus/Explicit

Stagnation pressure loads are defined by

$$p_s = -c_s \left(\mathbf{v} \cdot \mathbf{n} - \mathbf{v}_{ref} \cdot \mathbf{n}\right)^2,$$

where p_s is the stagnation pressure applied to the body; c_s is the factor, given as the magnitude of the load; $\mathbf{v}$ is the velocity of the point on the surface where the pressure is being applied; $\mathbf{n}$ is the unit outward normal to the element at the same point; and $\mathbf{v}_{ref}$ is the velocity of the reference node. The coefficient c_s should be very small to avoid excessive damping and a dramatic drop in the stable time increment.

Input File Usage: Use one of the following options to define a stagnation pressure load:

∗DLOAD, REF NODE=*reference_node*
element number or element set, SP*n* or SP, *magnitude*
∗DSLOAD, REF NODE=*reference_node*
element number or element set, SP, *magnitude*

Abaqus/CAE Usage: Use the following input to define a surface-based stagnation pressure load:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pressure** for the **Types for Selected Step**: **Distribution: Stagnation**, toggle on or off **Determine velocity from reference point**

Element-based stagnation pressure loads are not supported in Abaqus/CAE.

Pressure on pipe and elbow elements

You can specify external pressure, internal pressure, external hydrostatic pressure, or internal hydrostatic pressure on pipe or elbow elements. When pressure loads are applied, the effective outer or inner diameter must be specified in the element-based distributed load definition.

The loads resulting from the pressure on the ends of the element are included: Abaqus assumes a closed-end condition. Closed-end conditions correctly model the loading at pipe intersections, tight bends, corners, and cross-section changes; in straight sections and smooth bends the end loads of adjacent elements cancel each other precisely. If an open-end condition is to be modeled, a compensating point load should be added at the open end. A case where such an end load must be applied occurs if a pressurized pipe is modeled with a mixture of pipe and beam elements. In that case closed-end conditions generate a physically non-existing force at the transition between pipe and beam elements. Such mixed modeling of a pipe is not recommended.

For pipe elements subjected to pressure loading, the effective axial force due to the pressure loads can be obtained by requesting output variable ESF1 (see "Beam element library," Section 28.3.8).

Input File Usage: Use the following option to define an external pressure load on pipe or elbow elements:

∗DLOAD
element number or element set, PE or PENU, *magnitude*, *effective outer diameter*

Use the following option to define an internal pressure load on pipe or elbow elements:

∗DLOAD
element number or element set, PI or PINU, *magnitude*, *effective inner diameter*

Use the following option to define an external hydrostatic pressure load on pipe or elbow elements:

∗DLOAD
element number or element set, HPE, *magnitude*, *effective outer diameter*

Use the following option to define an internal hydrostatic pressure load on pipe or elbow elements:

∗DLOAD
element number or element set, HPI, *magnitude*, *effective inner diameter*

Abaqus/CAE Usage: Use the following input to define an external or internal pressure load on pipe or elbow elements:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pipe pressure** for the **Types for Selected Step**: **Side**: **External** or **Internal**, **Distribution**: **Uniform**, **User-defined**, or select an analytical field

Use the following input to define an external or internal hydrostatic pressure load on pipe or elbow elements:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Pipe pressure** for the **Types for Selected Step**: **Side**: **External** or **Internal**, **Distribution**: **Hydrostatic**

Defining distributed surface loads on plane stress elements

Plane stress theory assumes that the volume of a plane stress element remains constant in a large-strain analysis. When a distributed surface load is applied to an edge of plane stress elements, the current length and orientation of the edge are considered in the load distribution, but the current thickness is not; the original thickness is used.

This limitation can be circumvented only by using three-dimensional elements at the edge so that a change in thickness upon loading is recognized; suitable equation constraints ("Linear constraint equations," Section 33.2.1) would be required to make the in-plane displacements on the two faces of these elements equal. Three-dimensional elements along an edge can be connected to interior shell elements by using a shell-to-solid coupling constraint (see "Shell-to-solid coupling," Section 33.3.3, for details).

Edge tractions and moments on shell elements and line loads on beam elements

Distributed edge tractions (general, shear, normal, or transverse) and edge moments can be applied to shell elements in Abaqus as element-based or surface-based distributed loads. The units of an edge traction are force per unit length. The units of an edge moment are torque per unit length. References to local coordinate systems are ignored for all edge tractions and moments except general edge tractions.

Distributed line loads can be applied to beam elements in Abaqus as element-based distributed loads. The units of a line load are force per unit length.

Table 32.4.3–4 lists all of the distributed edge and line load types that are available in Abaqus, along with the corresponding load type labels. Part VI, "Elements," lists the distributed edge and line load types that are available for particular elements and the Abaqus/CAE load support for each load type. For element-based loads applied to shell elements, you must identify the edge of the element upon which the load is prescribed in the load type label (for example, EDLD*n* or EDLD*n*NU).

Follower edge and line loads

By definition, the line of action of a *follower* edge or line load rotates with the edge or line in a geometrically nonlinear analysis. This is in contrast to a *non-follower* load, which always acts in a fixed global direction.

With the exception of general edge tractions on shell elements and the forces per unit length in the global directions on beam elements, all the edge and line loads listed in Table 32.4.3–4 are modeled as follower loads. The normal, shear, and transverse edge loads listed in Table 32.4.3–4 act in the normal, shear, and transverse directions, respectively, in the current configuration (see Figure 32.4.3–6). The edge moment always acts about the shell edge in the current configuration. The forces per unit length in the local beam directions rotate with the beam elements.

Table 32.4.3–4 Distributed edge load types.

Load description	**Load type label for element-based loads**	**Load type label for surface-based loads**	**Abaqus/CAE load type**
General edge traction	EDLD*n*	EDLD	**Shell edge load**
Normal edge traction	EDNOR*n*	EDNOR	
Shear edge traction	EDSHR*n*	EDSHR	
Transverse edge traction	EDTRA*n*	EDTRA	
Edge moment	EDMOM*n*	EDMOM	
Nonuniform general edge traction	EDLD*n*NU	EDLDNU	**Shell edge load** (surface-based loads only)
Nonuniform normal edge traction	EDNOR*n*NU	EDNORNU	
Nonuniform shear edge traction	EDSHR*n*NU	EDSHRNU	
Nonuniform transverse edge traction	EDTRA*n*NU	EDTRANU	
Nonuniform edge moment	EDMOM*n*NU	EDMOMNU	
Force per unit length in global *X*-, *Y*-, and *Z*-directions (only for beam elements)	PX, PY, PZ	N/A	**Line load**
Nonuniform force per unit length in global *X*-, *Y*-, and *Z*-directions (only for beam elements)	PXNU, PYNU, PZNU	N/A	
Force per unit length in beam local 1- and 2-directions (only for beam elements)	P1, P2	N/A	
Nonuniform force per unit length in beam local 1- and 2-directions (only for beam elements)	P1NU, P2NU	N/A	

The forces per unit length in the global directions on beam elements are always non-follower loads.

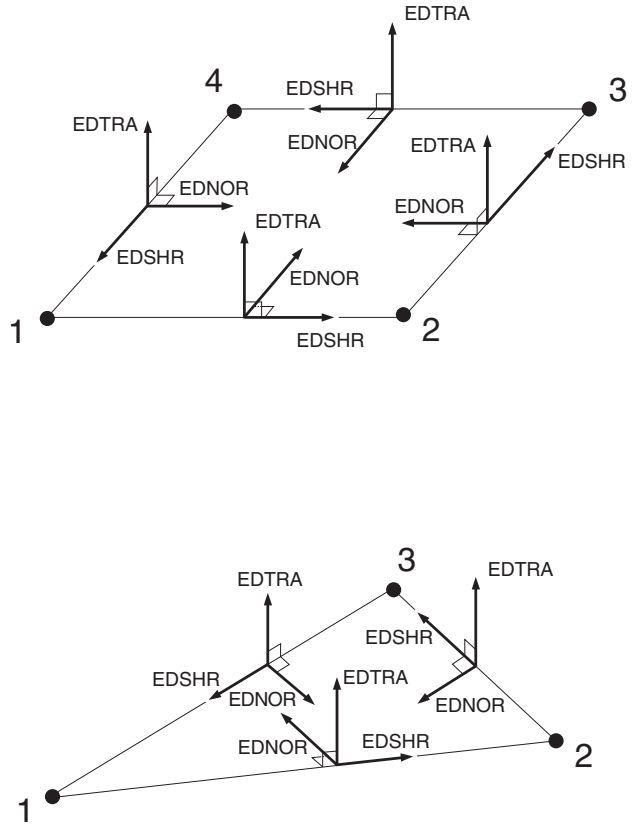


Figure 32.4.3–6 Positive edge loads.

General edge tractions can be specified to be follower or non-follower loads. There is no difference between a follower and a non-follower load in a geometrically linear analysis since the configuration of the body remains fixed.

Input File Usage: Use one of the following options to define general edge tractions as follower loads (the default):

∗DLOAD, FOLLOWER=YES
∗DSLOAD, FOLLOWER=YES

Use one of the following options to define general edge tractions as non-follower loads:

∗DLOAD, FOLLOWER=NO
∗DSLOAD, FOLLOWER=NO

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction: General**, toggle on or off **Follow rotation**

Specifying general edge tractions

General edge tractions allow you to specify an edge load, $\mathbf{t}$, acting on a shell edge, L. The resultant load, $\mathbf{f}$, is computed by integrating $\mathbf{t}$ over L:

$$\mathbf{f} = \int_L \mathbf{t}\, dL.$$

To define a general edge traction, you must provide both a magnitude, α, and direction, $\mathbf{t}_{user}$, for the load. The specified load directions are normalized by Abaqus; thus, they do not contribute to the magnitude of the load.

If a nonuniform general edge traction is specified, the magnitude, α, and direction, $\mathbf{t}_{user}$, must be specified in user subroutine **UTRACLOAD**.

Input File Usage: Use one of the following options to define a general edge traction:

*DLOAD
element number or element set, EDLD*n* or EDLD*n*NU, *magnitude*, *direction components*
*DSLOAD
surface name, EDLD or EDLDNU, *magnitude*, *direction components*

Abaqus/CAE Usage: Use the following input to define an element-based general edge traction:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction: General**, **Distribution**: select an analytical field

Use the following input to define a surface-based general edge traction:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction: General**, **Distribution**: **Uniform** or **User-defined**

Nonuniform element-based general edge traction is not supported in Abaqus/CAE.

Rotation of the load vector

In a geometrically linear analysis the edge load, $\mathbf{t}$, acts in the fixed direction defined by

$$\mathbf{t} = \alpha \frac{\mathbf{t}_{user}}{\|\mathbf{t}_{user}\|}.$$

If a non-follower load is specified in a geometrically nonlinear analysis (which includes a perturbation step about a geometrically nonlinear base state), the edge load, $\mathbf{t}$, acts in the fixed direction defined by

$$\mathbf{t} = \alpha \frac{\mathbf{t}_{user}}{||\mathbf{t}_{user}||}.$$

If a follower load is specified in a geometrically nonlinear analysis (which includes a perturbation step about a geometrically nonlinear base state), the components must be defined with respect to the reference configuration. The reference edge traction is defined as

$$\mathbf{t}^o = \alpha \frac{\mathbf{t}_{user}}{||\mathbf{t}_{user}||}.$$

The applied edge traction, $\mathbf{t}$, is computed by rigidly rotating $\mathbf{t}^o$ onto the current edge.

Defining the direction vector with respect to a local coordinate system

By default, the components of the edge traction vector are specified with respect to the global directions. You can also refer to a local coordinate system (see "Orientations," Section 2.2.5) for the direction components of these tractions.

Input File Usage: Use one of the following options to specify a local coordinate system:

∗DLOAD, ORIENTATION=*name*
∗DSLOAD, ORIENTATION=*name*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: select **CSYS: Picked** and click **Edit** to pick a local coordinate system, or select **CSYS: User-defined** to enter the name of a user subroutine that defines a local coordinate system

Specifying shear, normal, and transverse edge tractions

The loading directions of shear, normal, and transverse edge tractions are determined by the underlying elements. A positive shear edge traction acts in the positive direction of the shell edge as determined by the element connectivity. A positive normal edge traction acts in the plane of the shell in the inward direction. A positive transverse edge traction acts in a sense opposite to the facet normal. The directions of positive shear, normal, and transverse edge tractions are shown in Figure 32.4.3–6.

To define a shear, normal, or transverse edge traction, you must provide a magnitude, α for the load.

If a nonuniform shear, normal, or transverse edge traction is specified, the magnitude, α, must be specified in user subroutine **UTRACLOAD**.

In a geometrically linear step, the shear, normal, and transverse edge tractions act in the tangential, normal, and transverse directions of the shell, as shown in Figure 32.4.3–6. In a geometrically nonlinear analysis the shear, normal, and transverse edge tractions rotate with the shell edge so they always act in the tangential, normal, and transverse directions of the shell, as shown in Figure 32.4.3–6.

Input File Usage: Use one of the following options to define a directed edge traction:

∗DLOAD
element number or element set, *directed edge traction label*, *magnitude*
∗DSLOAD
surface name, *directed edge traction label*, *magnitude*

For element-based loads the *directed edge traction label* can be EDSHR*n* or EDSHR*n*NU for shear edge tractions, EDNOR*n* or EDNOR*n*NU for normal edge tractions, or EDTRA*n* or EDTRA*n*NU for transverse edge tractions.

For surface-based loads the *directed edge traction label* can be EDSHR or EDSHRNU for shear edge tractions, EDNOR or EDNORNU for normal edge tractions, or EDTRA or EDTRANU for transverse edge tractions.

Abaqus/CAE Usage: Use the following input to define an element-based directed edge traction:

Load module: **Create Load**; choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**; **Traction: Normal**, **Transverse**, or **Shear**; **Distribution**: select an analytical field

Use the following input to define a surface-based directed edge traction:

Load module: **Create Load**; choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**; **Traction: Normal**, **Transverse**, or **Shear**; **Distribution**: **Uniform** or **User-defined**

Nonuniform element-based directed edge traction is not supported in Abaqus/CAE.

Specifying edge moments

An edge moment acts about the shell edge with the positive direction determined by the element connectivity. The directions of positive edge moments are shown in Figure 32.4.3–7.

To define a distributed edge moment, you must provide a magnitude, α, for the load.

If a nonuniform edge moment is specified, the magnitude, α, must be specified in user subroutine **UTRACLOAD**.

An edge moment always acts about the current shell edge in both geometrically linear and nonlinear analyses.

In a geometrically linear step an edge moment acts about the shell edge as shown in Figure 32.4.3–7. In a geometrically nonlinear analysis an edge moment always acts about the shell edge as shown in Figure 32.4.3–7.

Input File Usage: Use one of the following options to define an edge moment:

∗DLOAD
element number or element set, EDMOM*n* or EDMOM*n*NU, *magnitude*
∗DSLOAD
surface name, EDMOM or EDMOMNU, *magnitude*

Abaqus/CAE Usage: Use the following input to define an element-based edge moment:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction: Moment**, **Distribution**: select an analytical field

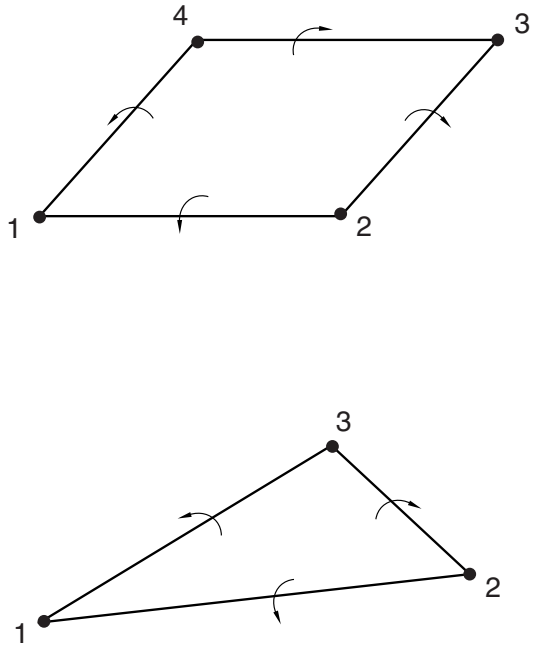


Figure 32.4.3–7 Positive edge moments.

Use the following input to define a surface-based edge moment:

Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction: General**, **Distribution**: **Uniform** or **User-defined**

Nonuniform element-based edge moments are not supported in Abaqus/CAE.

Resultant loads due to edge tractions and moments

You can choose to integrate edge tractions and moments over the current or the reference configuration by specifying whether or not a constant resultant should be maintained. In general, the constant resultant method is best suited for cases where the magnitude of the resultant load should not vary with changes in the edge length. However, it is up to you to decide which approach is best for your analysis.

Choosing not to have a constant resultant

If you choose not to have a constant resultant, an edge traction or moment is integrated over the edge in the current configuration, an edge whose length changes during a geometrically nonlinear analysis.

Input File Usage: Use one of the following options:

*DLOAD, CONSTANT RESULTANT=NO
*DSLOAD, CONSTANT RESULTANT=NO

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction is defined per unit deformed area**

Maintaining a constant resultant

If you choose to have a constant resultant, an edge traction or moment is integrated over the edge in the reference configuration, whose length is constant.

Input File Usage: Use one of the following options:

∗DLOAD, CONSTANT RESULTANT=YES
∗DSLOAD, CONSTANT RESULTANT=YES

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Shell edge load** for the **Types for Selected Step**: **Traction is defined per unit undeformed area**

Specifying line loads on beam elements

You can specify line loads on beam elements in the global X-, Y-, or Z-direction. In addition, you can specify line loads on beam elements in the beam local 1- or 2-direction.

Input File Usage: Use the following option to define a force per unit length in the global X-, Y-, or Z-direction on beam elements:

∗DLOAD
element number or element set, *load type label*, *magnitude*

where *load type label* is PX, PY, PZ, PXNU, PYNU, or PZNU.

Use the following option to define a force per unit length in the beam local 1- or 2-direction:

∗DLOAD
element number or element set, *load type label*, *magnitude*

where *load type label* is P1, P2, P1NU, or P2NU.

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Line load** for the **Types for Selected Step**

32.4.4 THERMAL LOADS

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CFD Abaqus/CAE

References

- "Applying loads: overview," Section 32.4.1
- *CFLUX
- *DFLUX
- *DSFLUX
- *CFILM
- *FILM
- *SFILM
- *FILM PROPERTY
- *CRADIATE
- *RADIATE
- *SRADIATE
- "Defining a concentrated heat flux," Section 16.9.19 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a body heat flux," Section 16.9.18 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a surface heat flux," Section 16.9.17 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a fluid wall boundary condition," Section 16.10.10 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a surface film condition interaction," Section 15.13.19 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a concentrated film condition interaction," Section 15.13.20 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a surface radiative interaction," Section 15.13.21 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a concentrated radiative interaction," Section 15.13.22 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Thermal loads can be applied in heat transfer analysis, in fully coupled temperature-displacement analysis, fully coupled thermal-electrical-structural analysis, and in coupled thermal-electrical analysis,

as outlined in "Prescribed conditions: overview," Section 32.1.1. The following types of thermal loads are available:

- Concentrated heat flux prescribed at nodes.
- Distributed heat flux prescribed on element faces or surfaces.
- Body heat flux per unit volume.
- Boundary convection defined at nodes, on element faces, or on surfaces.
- Boundary radiation defined at nodes, on element faces, or on surfaces.

See "Applying loads: overview," Section 32.4.1, for general information that applies to all types of loading.

Modeling thermal radiation

The following types of radiation heat exchange can be modeled using Abaqus:

- Exchange between a nonconcave surface and a nonreflecting environment. This type of radiation is modeled using boundary radiation loads defined at nodes, on element faces, or on surfaces, as described below.
- Exchange between two surfaces within close proximity of each other in which temperature gradients along the surfaces are not large. This type of radiation is modeled using the gap radiation capability described in "Thermal contact properties," Section 35.2.1.
- Exchange between surfaces that constitute a cavity. This type of radiation is modeled using the cavity radiation capability available in Abaqus/Standard and described in "Cavity radiation," Section 39.1.1, or through the average-temperature radiation condition described in "Specifying average-temperature radiation conditions," below.

Prescribing heat fluxes directly

Concentrated heat fluxes can be prescribed at nodes (or node sets). Distributed heat fluxes can be defined on element faces or surfaces.

Specifying concentrated heat fluxes

By default, a concentrated heat flux is applied to degree of freedom 11. For shell heat transfer elements concentrated heat fluxes can be prescribed through the thickness of the shell by specifying degree of freedom 11, 12, 13, etc. Temperature variation through the thickness of shell elements is described in "Choosing a shell element," Section 28.6.2.

Input File Usage: *CFLUX
node number or node set name, *degree of freedom*, *heat flux magnitude*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Thermal** for the **Category** and **Concentrated heat flux** for the **Types for Selected Step**: select region: **Magnitude**: *heat flux magnitude*

Defining the values of concentrated nodal flux from a user-specified file

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

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

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

Specifying element-based distributed heat fluxes

You can specify element-based distributed surface fluxes (on element faces) or body fluxes (flux per unit volume). For surface fluxes you must identify the face of the element upon which the flux is prescribed in the flux label (for example, S*n* or S*n*NU for continuum elements). The distributed flux types available depend on the element type. Part VI, "Elements," lists the distributed fluxes that are available for particular elements.

Input File Usage: *DFLUX
element number or element set name, *load type label*, *flux magnitude*

where *load type label* is S*n*, SPOS, SNEG, S1, S2, or BF

Abaqus/CAE Usage: Use the following input to define a distributed surface flux:

Load module: **Create Load**: choose **Thermal** for the **Category** and **Surface heat flux** for the **Types for Selected Step**: select region: **Distribution**: select an analytical field, **Magnitude**: *flux magnitude*

Use the following input to define a distributed body flux:

Load module: **Create Load**: choose **Thermal** for the **Category** and **Body heat flux** for the **Types for Selected Step**: select region: **Distribution**: **Uniform** or select an analytical field, **Magnitude**: *flux magnitude*

You can specify only a uniform distribution in an Abaqus/CFD analysis.

Specifying surface-based distributed heat fluxes

When you specify distributed surface fluxes on a surface, the surface that contains the element and face information is defined as described in "Element-based surface definition," Section 2.3.2. You must specify the surface name, the heat flux label, and the heat flux magnitude.

Input File Usage: *DSFLUX
surface name, S, *flux magnitude*

Abaqus/CAE Usage: Use the following input to specify surface-based distributed heat fluxes:

Load module: **Create Load**: choose **Thermal** for the **Category** and **Surface heat flux** for the **Types for Selected Step**: select region: **Distribution**: **Uniform**, **Magnitude**: *flux magnitude*

Use the following input to specify surface-based distributed wall heat fluxes in Abaqus/CFD:

Load module: **Create Boundary Condition**: **Step**: ***flow_step***: choose **Fluid** for the **Category** and **Fluid wall condition** for the **Types for Selected Step**: select region: **Thermal Energy**: **Specify**: **Heat flux**, **Magnitude**: *flux magnitude*

Modifying or removing heat fluxes

Heat fluxes can be added, modified, or removed as described in "Applying loads: overview," Section 32.4.1.

Specifying time-dependent heat fluxes

The magnitude of a concentrated or a distributed heat flux can be controlled by referring to an amplitude curve. If different magnitude variations are needed for different fluxes, the flux definitions can be repeated, with each referring to its own amplitude curve. See "Prescribed conditions: overview," Section 32.1.1, and "Amplitude curves," Section 32.1.2, for details.

Defining nonuniform distributed heat flux in a user subroutine

In Abaqus/Standard a nonuniform distributed flux (element-based or surface-based) can be defined in user subroutine **DFLUX**. The specified reference magnitude will be passed into user subroutine **DFLUX** as **FLUX(1)**. If the magnitude is omitted, **FLUX(1)** will be passed in as zero.

Input File Usage: Use the following option to define a nonuniform element-based heat flux:

*DFLUX
element number or element set name, *load type label*, *flux magnitude*

where *load type label* is S*n*NU, SPOSNU, SNEGNU, S1NU, S2NU, or BFNU.

Use the following option to define a nonuniform surface-based heat flux:

*DSFLUX
surface name, SNU, *flux magnitude*

For example, for general heat transfer shell elements ("Three-dimensional conventional shell element library," Section 28.6.7) a uniform surface flux of 10.0 per unit area on the top face (SPOS) of shell element 100 can be applied by

```
*DFLUX
 100, SPOS, 10.0
```

When the variation of the (nonuniform) flux magnitude is defined by means of user subroutine **DFLUX**, the distributed flux type label SPOSNU is used.

```
*DFLUX
 100, SPOSNU, magnitude
```

Abaqus/CAE Usage: Use the following input to define a nonuniform element-based body flux:

Load module: **Create Load**: choose **Thermal** for the **Category** and **Body heat flux** for the **Types for Selected Step**: select region: **Distribution: User-defined**, **Magnitude**: *flux magnitude*

Use the following input to define a nonuniform surface-based heat flux:

Load module: **Create Load**: choose **Thermal** for the **Category** and **Surface heat flux** for the **Types for Selected Step**: select region: **Distribution: User-defined**, **Magnitude**: *flux magnitude*

Nonuniform element-based distributed surface fluxes are not supported in Abaqus/CAE.

Prescribing boundary convection

Heat flux on a surface due to convection is governed by

$$q = -h(\theta - \theta^0),$$

where

- q is the heat flux across the surface,
- h is a reference film coefficient,
- θ is the temperature at this point on the surface, and
- θ^0 is a reference sink temperature value.

Heat flux due to convection can be defined on element faces, on surfaces, or at nodes.

Specifying element-based film conditions

You can define the sink temperature value, θ^0, and the film coefficient, h, on element faces. The convection is applied to element edges in two dimensions and to element faces in three dimensions. The edge or face of the element upon which the film is placed is identified by a film load type label and depends on the element type (see Part VI, "Elements"). You must specify the element number or element set name, the film load type label, a sink temperature, and a film coefficient.

Input File Usage: *FILM
element number or element set name, *film load type label*, θ^0, h

Abaqus/CAE Usage: Element-based film conditions are supported in Abaqus/CAE only for the film coefficient.

Interaction module: **Create Interaction**: **Surface film condition**: select region: **Definition**: select an analytical field: **Film coefficient:** h

Specifying surface-based film conditions

You can define the sink temperature value, θ^0, and the film coefficient, h, on a surface. The surface that contains the element and face information is defined as described in "Element-based surface definition," Section 2.3.2. You must specify the surface name, the film load type, a sink temperature, and a film coefficient.

Input File Usage: ∗SFILM
surface name, F or FNU, θ^0, h

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface film condition**:
select region: **Definition**: **Embedded Coefficient** or **User-defined**:
Film coefficient: h and **Sink temperature:** θ^0

Specifying node-based film conditions

A node-based film condition requires that you define the nodal area for a specified node number or node set; the sink temperature value, θ^0; and the film coefficient, h. The associated degree of freedom is 11. For shell type elements where the film is associated with a degree of freedom other than 11, you can specify the concentrated film for a duplicate node that is constrained to the appropriate degree of freedom of the shell node by using an equation constraint (see "Linear constraint equations," Section 33.2.1).

Input File Usage: ∗CFILM
node number or node set name, *nodal area*, θ^0, h

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Concentrated film condition**:
select region: **Definition**: **Embedded Coefficient**, **User-defined**,
or select an analytical field: **Associated nodal area:** *nodal area*,
Film coefficient: h, **Sink temperature:** θ^0

Specifying temperature- and field-variable-dependent film conditions

If the film coefficient is a function of temperature, you can specify the film property data separately and specify the name of the property table instead of the film coefficient in the film condition definition.

You can specify multiple film property tables to define different variations of the film coefficient, h, as a function of surface temperature and/or field variables. Each film property table must be named. This name is referred to by the film condition definitions.

A new film property table can be defined in a restart step. If a film property table with an existing name is encountered, the second definition is ignored.

Input File Usage: For element-based film conditions, use the following options:

∗FILM PROPERTY, NAME=*film property table name*
∗FILM
element number or element set name, *film load type label*,
θ^0, *film property table name*

For surface-based film conditions, use the following options:

∗FILM PROPERTY, NAME=*film property table name*

∗SFILM
surface name, F, θ^0, *film property table name*

For node-based film conditions, use the following options:

∗FILM PROPERTY, NAME=*film property table name*
∗CFILM
node number or node set name, *nodal area*, θ^0, *film property table name*

The ∗FILM PROPERTY option must appear in the model definition portion of the input file.

Abaqus/CAE Usage: Interaction module:
Create Interaction Property: **Name**: *film property table name* and **Film condition**
Create Interaction: **Surface film condition** or **Concentrated film condition**: select region: **Definition: Property Reference** and **Film interaction property**: *film property table name*

Modifying or removing film conditions

Film conditions can be added, modified, or removed as described in "Applying loads: overview," Section 32.4.1.

Specifying time-dependent film conditions

For a uniform film both the sink temperature and the film coefficient can be varied with time by referring to amplitude definitions. One amplitude curve defines the variation of the sink temperature, θ^0, with time. Another amplitude curve defines the variation of the film coefficient, h, with time. See "Prescribed conditions: overview," Section 32.1.1, and "Amplitude curves," Section 32.1.2, for more information.

Input File Usage: Use the following options to define time-dependent film conditions:

∗AMPLITUDE, NAME=*temp_amp*
∗AMPLITUDE, NAME=*h_amp*
∗FILM, AMPLITUDE=*temp_amp*, FILM AMPLITUDE=*h_amp*
∗SFILM, AMPLITUDE=*temp_amp*, FILM AMPLITUDE=*h_amp*
∗CFILM, AMPLITUDE=*temp_amp*, FILM AMPLITUDE=*h_amp*

Abaqus/CAE Usage: Use the following input to define time-dependent film conditions. If you select an analytical field to define the interaction, the analytical field affects only the film coefficient.

Interaction module:
Create Amplitude: **Name:** *h_amp*
Create Amplitude: **Name:** *temp_amp*
Create Interaction: **Surface film condition** or **Concentrated film condition**: select region: **Definition**: **Embedded Coefficient** or select an analytical field: **Film coefficient amplitude**: *h_amp* and **Sink amplitude**: *temp_amp*

Examples

A uniform, time-dependent film condition can be defined for face 2 of element 3 by

```
*AMPLITUDE, NAME=sink
 0.0, 0.5, 1.0, 0.9
*AMPLITUDE, NAME=famp
 0.0, 1.0, 1.0, 22.0
 …
*STEP
** For an Abaqus/Standard analysis:
*HEAT TRANSFER
** For an Abaqus/Explicit analysis:
*DYNAMIC TEMPERATURE-DISPLACEMENT, EXPLICIT
 …
*FILM, AMPLITUDE=sink, FILM AMPLITUDE=famp
 3, F2, 90.0, 2.0
```

A uniform, temperature-dependent film coefficient and a time-dependent sink temperature can be defined for face 2 of element 3 by

```
*AMPLITUDE, NAME=sink
0.0, 0.5, 1.0, 0.9
*FILM PROPERTY, NAME=filmp
 2.0,  80.0
 2.3,  90.0
 8.5, 180.0
 …
*STEP
** For an Abaqus/Standard analysis:
*HEAT TRANSFER
** For an Abaqus/Explicit analysis:
*DYNAMIC TEMPERATURE-DISPLACEMENT, EXPLICIT
 …
*FILM, AMPLITUDE=sink
 3, F2, 90.0, filmp
```

A uniform, temperature-dependent film coefficient and a time-dependent sink temperature can be defined for node 2, where the nodal area is 50, by

```
*AMPLITUDE, NAME=sink
0.0, 0.5, 1.0, 0.9
*FILM PROPERTY, NAME=filmp
 2.0,  80.0
 2.3,  90.0
```

```
 8.5, 180.0
 …
*STEP
** For an Abaqus/Standard analysis:
*HEAT TRANSFER
** For an Abaqus/Explicit analysis:
*DYNAMIC TEMPERATURE-DISPLACEMENT, EXPLICIT
 …
*CFILM, AMPLITUDE=sink,
 2, 50, 90.0, filmp
```

Defining nonuniform film conditions in a user subroutine

In Abaqus/Standard a nonuniform film coefficient can be defined as a function of position, time, temperature, etc. in user subroutine **FILM** for element-based, surface-based, as well as node-based film conditions. Amplitude references are ignored if a nonuniform film is prescribed.

Input File Usage: Use the following option to define a nonuniform film coefficient for an element-based film condition:

∗FILM
element number or element set name, F*n*NU

Use the following option to define a nonuniform film coefficient for a surface-based film condition:

∗SFILM
surface name, FNU

Use the following option to define a nonuniform film coefficient for a node-based film condition:

∗CFILM, USER
node number or node set name, *nodal area*

Abaqus/CAE Usage: Element-based film conditions to define a nonuniform film coefficient are not supported in Abaqus/CAE. However, similar functionality is available using surface-based film conditions. Use the following option to define a nonuniform film coefficient for a surface-based film condition:

Interaction module: **Create Interaction**: **Surface film condition**:
select region: **Definition**: **User-defined**

Use the following option to define a nonuniform film coefficient for a node-based film condition:

Interaction module: **Create Interaction**: **Concentrated film condition**:
select region: **Definition**: **User-defined**

Prescribing boundary radiation

Heat flux on a surface due to radiation to the environment is governed by

$$q = \sigma\epsilon\left[(\theta - \theta^Z)^4 - (\theta^0 - \theta^Z)^4\right],$$

where

- q is the heat flux across the surface,
- ϵ is the emissivity of the surface,
- σ is the Stefan-Boltzmann constant,
- θ is the temperature at this point on the surface,
- θ^0 is an ambient temperature value, and
- θ^Z is the value of absolute zero on the temperature scale being used.

Heat flux due to radiation can be defined on element faces, on surfaces, or at nodes.

Specifying element-based radiation

To specify element-based radiation within a heat transfer or coupled temperature-displacement step definition, you must provide the ambient temperature value, θ^0, and the emissivity of the surface, ϵ. The radiation is applied to element edges in two dimensions and to element faces in three dimensions. The edge or face of the element upon which the radiation occurs is identified by a radiation type label depending on the element type (see Part VI, "Elements").

Input File Usage: *RADIATE
element number or element set name, R*n*, θ^0, ϵ

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface radiation**: select region: **Radiation type: To ambient**, **Emissivity distribution:** select an analytical field, **Emissivity:** ϵ, and **Ambient temperature:** θ^0

Specifying surface-based radiation to ambient

You can apply the radiation to a surface rather than to individual element faces. The surface that contains the element and face information is defined as described in "Element-based surface definition," Section 2.3.2. You must specify the surface name; the radiation load type label, R (or RPOS, RNEG in the case of shells); the ambient temperature value, θ^0; and the emissivity of the surface, ϵ.

Input File Usage: *SRADIATE
surface name, R, θ^0, ϵ

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface radiation**: select region: **Radiation type: To ambient**, **Emissivity distribution: Uniform**, **Emissivity:** ϵ, and **Ambient temperature:** θ^0

Specifying node-based radiation to ambient

To specify node-based radiation within a heat transfer or coupled temperature-displacement step definition, you must provide the nodal area for a specified node number or node set; the ambient temperature value, θ^0; and the emissivity of the surface, ϵ. The associated degree of freedom is 11. For shell elements where the concentrated radiation is associated with a degree of freedom other than 11, you can specify the required data for a duplicate node that is constrained to the appropriate degree of freedom of the shell node by using an equation constraint.

Input File Usage: *CRADIATE
node number or node set name, *nodal area*, θ^0, ϵ

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Concentrated radiation to ambient**: select region: **Associated nodal area: Emissivity:** ϵ and **Ambient temperature:** θ^0

Specifying time-dependent radiation

The user-specified value of the ambient temperature, θ^0, can be varied throughout the step by referring to an amplitude definition. See "Applying loads: overview," Section 32.4.1, and "Amplitude curves," Section 32.1.2, for details.

Specifying average-temperature radiation conditions

The average-temperature radiation condition is an approximation to the cavity radiation problem, where the radiative flux per unit area into a facet is

$$q_i^c = \sigma \epsilon_i \left(\theta_{AVG}^4 - (\theta_i - \theta^Z)^4 \right),$$

with the average temperature for the surface θ_{AVG} being calculated as

$$\theta_{AVG}^4 = \frac{1}{A_{total}} \sum_{j=1}^{N} A_j \, (\theta_j - \theta^Z)^4.$$

The average temperature in the cavity is computed at the beginning of each increment and held constant over the increment. Therefore, the average-temperature radiation condition has some dependency on the increment size, and you need to ensure that the increment size you use is appropriate for your model. If you see large changes in temperature over an increment, you may need to reduce the increment size.

Input File Usage: Use the following option to define the average-temperature radiation condition on a surface:

*SRADIATE
surface name, AVG, , ϵ

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface radiation**: select the surface region: **Radiation type: Cavity approximation (3D only)**, **Emissivity:** ϵ

Specifying the value of absolute zero

You can specify the value of absolute zero, θ^Z, on the temperature scale being used; you must specify this value as model data. By default, the value of absolute zero is 0.0.

Input File Usage: *PHYSICAL CONSTANTS, ABSOLUTE ZERO=θ^Z

Abaqus/CAE Usage: Any module: **Model→Edit Attributes→*model_name***:
Absolute zero temperature: θ^Z

Specifying the value of the Stefan-Boltzmann constant

If boundary radiation is prescribed, you must specify the Stefan-Boltzmann constant, σ; this value must be specified as model data.

Input File Usage: *PHYSICAL CONSTANTS, STEFAN BOLTZMANN=σ

Abaqus/CAE Usage: Any module: **Model→Edit Attributes→*model_name***:
Stefan-Boltzmann constant: σ

Modifying or removing boundary radiation

Boundary radiation conditions can be added, modified, or removed as described in "Applying loads: overview," Section 32.4.1.

32.4.5 ELECTROMAGNETIC LOADS

Products: Abaqus/Standard Abaqus/CAE

References

- "Prescribed conditions: overview," Section 32.1.1
- "Applying loads: overview," Section 32.4.1
- *CECHARGE
- *CECURRENT
- *DECHARGE
- *DECURRENT
- *DSECHARGE
- *DSECURRENT
- "Defining a concentrated current," Section 16.9.24 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a surface current," Section 16.9.25 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a body current," Section 16.9.26 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a concentrated charge," Section 16.9.27 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a surface charge," Section 16.9.28 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a body charge," Section 16.9.29 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

As outlined in "Prescribed conditions: overview," Section 32.1.1, electromagnetic loads can be applied in "Piezoelectric analysis," Section 6.7.2; "Coupled thermal-electrical analysis," Section 6.7.3; "Fully coupled thermal-electrical-structural analysis," Section 6.7.4; and "Time-harmonic eddy current analysis," Section 6.7.5.

The types of electromagnetic loads available depend on the analysis being performed, as described in the sections below. See "Applying loads: overview," Section 32.4.1, for general information that applies to all types of loading.

Defining time-dependent electromagnetic loads

The prescribed magnitude of a concentrated or a distributed electromagnetic load can vary with time during a step according to an amplitude definition, as described in "Prescribed conditions: overview," Section 32.1.1. If different variations are needed for different loads, each load can refer to its own amplitude definition.

In an eddy current analysis all loads are assumed to be time-harmonic.

Modifying electromagnetic loads

Concentrated or a distributed electromagnetic loads can be added, modified, or removed as described in "Applying loads: overview," Section 32.4.1.

Prescribing electromagnetic loads for piezoelectric analyses

In a piezoelectric analysis a concentrated electric charge can be prescribed at nodes, a distributed electric surface charge can be defined on element faces and surfaces, and a distributed electric body charge can be defined on elements.

Specifying concentrated electric charge

To specify a concentrated electric charge, specify the node or node set and the magnitude of the charge.

Input File Usage: *CECHARGE
node number or node set name, , charge magnitude

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical** for the **Category** and **Concentrated charge** for the **Types for Selected Step**; **Magnitude**: *charge magnitude*

Specifying element-based distributed electric charge

You can specify a distributed surface charge (on element faces) or a distributed body charge (charge per unit volume). For an element-based surface charge you must identify the face of the element upon which the charge is prescribed in the charge label. The distributed charge types available depend on the element type. Part VI, "Elements," lists the distributed charges that are available for particular elements.

Input File Usage: *DECHARGE
element number or element set name, charge label, charge magnitude

where *charge label* is ES*n* or EBF

Abaqus/CAE Usage: Use the following input to define a distributed surface charge on element faces:

Load module: **Create Load**: choose **Electrical** for the **Category** and **Surface charge** for the **Types for Selected Step**; **Distribution**: select an analytical field, **Magnitude**: *charge magnitude*

Use the following input to define a body charge:

Load module: **Create Load**: choose **Electrical** for the **Category** and **Body charge** for the **Types for Selected Step**

Specifying surface-based distributed electric charge

When you specify a distributed electric charge on a surface, the element-based surface (see "Element-based surface definition," Section 2.3.2) contains the element and face information. You must specify the surface name, the electric charge label, and the electric charge magnitude.

Input File Usage: ∗DSECHARGE
surface name, ES, *charge magnitude*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical** for the **Category** and **Surface charge** for the **Types for Selected Step**; **Distribution**: **Uniform**, **Magnitude**: *charge magnitude*

Specifying electric charge in direct-solution steady-state dynamics analysis

In the direct-solution steady-state dynamics procedure, electric charges are given in terms of their real and imaginary components.

Input File Usage: Use the following options to define electric charges in direct-integration steady-state dynamics analysis:

∗CECHARGE, REAL or IMAGINARY *(real or imaginary component)*
∗DECHARGE, REAL or IMAGINARY
∗DSECHARGE, REAL or IMAGINARY

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical** for the **Category** and **Concentrated charge**, **Surface charge**, or **Body charge** for the **Types for Selected Step**; **Magnitude**: *real component + imaginary component*

Loading in mode-based and subspace-based procedures

Electrical charge loads should be used only in conjunction with residual modes in the eigenvalue extraction step, due to the "massless" mode effect. Since the electrical potential degrees of freedom do not have any associated mass, these degrees of freedom are essentially eliminated (similar to Guyan reduction or mass condensation) during the eigenvalue extraction. The residual modes represent the static response corresponding to the electrical charge loads, which will adequately represent the potential degree of freedom in the eigenspace.

Prescribing electromagnetic loads for coupled thermal-electrical and fully coupled thermal-electrical-structural analyses

In a coupled thermal-electrical analysis and fully coupled thermal-electrical-structural analysis a concentrated current can be prescribed at nodes, distributed current densities can be defined on element faces and surfaces, and distributed body currents can be defined on elements.

Specifying concentrated current density

To define concentrated currents, specify the node or node set and the magnitude of the current.

Input File Usage: *CECURRENT
node number or node set name, , current magnitude

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical** for the **Category** and **Concentrated current** for the **Types for Selected Step**; **Magnitude**: *current magnitude*

Specifying element-based distributed current density

You can specify distributed surface current densities (on element faces) or distributed body current densities (current per unit volume). For element-based surface current densities you must identify the face of the element upon which the current is prescribed in the current label. The distributed current types available depend on the element type. Part VI, "Elements," lists the distributed current densities that are available for particular elements.

Input File Usage: *DECURRENT
element number or element set name, current density label, current density magnitude

where *current density label* is CS*n*, CS1, CS2, or CBF

Abaqus/CAE Usage: Use the following input to define a distributed surface current density on element faces:

Load module: **Create Load**: choose **Electrical** for the **Category** and **Surface current** for the **Types for Selected Step**; **Distribution**: select an analytical field, **Magnitude**: *current density magnitude*

Use the following input to define a body current density:

Load module: **Create Load**: choose **Electrical** for the **Category** and **Body current** for the **Types for Selected Step**

Specifying surface-based distributed current densities

When you specify distributed current densities on a surface, the element-based surface (see "Element-based surface definition," Section 2.3.2) contains the element and face information. You must specify the surface name, the current density label, and the current density magnitude.

Input File Usage: *DSECURRENT
surface name, CS, *current density magnitude*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Electrical** for the **Category** and **Surface current** for the **Types for Selected Step**: **Distribution**: **Uniform**, **Magnitude**: *current density magnitude*

Prescribing electromagnetic loads for eddy current analyses

In an eddy current analysis a distributed surface current density vector can be defined on surfaces and a distributed body current density vector can be defined on elements.

Specifying element-based distributed current density vectors

When you define a distributed body current density vector, you must specify the element or element set, the current density vector label, the magnitude of the current density vector, the vector components of the current density, and an optional orientation name that defines the local coordinate system in which the vector components are specified. By default, the vector components of the current density are defined with respect to the global directions.

The specified current density vector direction components are normalized by Abaqus and, thus, do not contribute to the magnitude of the load.

Input File Usage: *DECURRENT
element number or element set name, CJ, *current density vector magnitude, current density vector direction components, orientation name*

Abaqus/CAE Usage: Current density vectors are not supported in Abaqus/CAE.

Specifying surface-based distributed current density vectors

When you specify distributed current density vectors on a surface, the element-based surface (see "Element-based surface definition," Section 2.3.2) contains the element and face information. You must specify the surface name, the current density vector label, and the magnitude of the current density vector, the vector components of the current density, and an optional orientation name that defines the local coordinate system in which the surface current density is specified. By default, the vector components of the current density are defined with respect to the global directions.

The specified current density vector direction components are normalized by Abaqus and, thus, do not contribute to the magnitude of the load.

Input File Usage: *DSECURRENT
surface name, CK, *current density vector magnitude, current density vector direction components, orientation name*

Abaqus/CAE Usage: Current density vectors are not supported in Abaqus/CAE.

Defining nonuniform current density vectors in a user subroutine

Nonuniform body current density vectors can be defined with user subroutine **UDECURRENT**, and nonuniform surface current density vectors can be defined with user subroutine **UDSECURRENT**. If the magnitude and direction components are given, the values are passed into the user subroutine.

Input File Usage: Use the following option to define nonuniform element-based current density vectors:

*DECURRENT
element number or element set name, CJNU, *current density vector magnitude, current density vector direction components, orientation name*

Use the following option to define nonuniform surface-based current density vectors:

*DSECURRENT
surface name, CKNU, *current density vector magnitude, current density vector direction components, orientation name*

Abaqus/CAE Usage: Current density vectors are not supported in Abaqus/CAE.

Specifying real and imaginary components of current density vectors

In an eddy current analysis, current density vectors are given in terms of their real and imaginary components.

Input File Usage: Use the following options to define current density vectors:

*DECURRENT, REAL or IMAGINARY
*DSECURRENT, REAL or IMAGINARY

Abaqus/CAE Usage: Current density vectors are not supported in Abaqus/CAE.

32.4.6 ACOUSTIC AND SHOCK LOADS

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Applying loads: overview," Section 32.4.1
- "Acoustic, shock, and coupled acoustic-structural analysis," Section 6.10.1
- *AMPLITUDE
- *BOUNDARY
- *CLOAD
- *CONWEP CHARGE PROPERTY
- *IMPEDANCE
- *IMPEDANCE PROPERTY
- *INCIDENT WAVE
- *INCIDENT WAVE FLUID PROPERTY
- *INCIDENT WAVE INTERACTION
- *INCIDENT WAVE INTERACTION PROPERTY
- *INCIDENT WAVE PROPERTY
- *INCIDENT WAVE REFLECTION
- *SIMPEDANCE
- *UNDEX CHARGE PROPERTY
- "Defining acoustic impedance," Section 15.13.14 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining incident waves," Section 15.13.15 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining an acoustic impedance interaction property," Section 15.14.4 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining an incident wave interaction property," Section 15.14.5 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Acoustic loads can be applied only in transient or steady-state dynamic analysis procedures. The following types of acoustic loads are available:

- Boundary impedance defined on element faces or on surfaces.
- Nonreflecting radiation boundaries in exterior problems such as a structure vibrating in an acoustic medium of infinite extent.

- Concentrated pressure-conjugate loads prescribed at acoustic element nodes.
- Temporally and spatially varying pressure loading on acoustic and solid surfaces due to incident waves traveling through the acoustic medium.

Specified boundary impedance

A boundary impedance specifies the relationship between the pressure of an acoustic medium and the normal motion at the boundary. Such a condition is applied, for example, to include the effect of small-amplitude "sloshing" in a gravity field or the effect of a compressible, possibly dissipative, lining (such as a carpet) between an acoustic medium and a fixed, rigid wall or structure.

The impedance boundary condition at any point along the acoustic medium surface is governed by

$$\dot{u}_{out} = \frac{1}{k_1}\dot{p} + \frac{1}{c_1}p,$$

where

$\dot{u}_{out}$ is the acoustic particle velocity in the outward normal direction of the acoustic medium surface,

p is the acoustic pressure,

$\dot{p}$ is the time rate of change of the acoustic pressure,

$1/k_1$ is the proportionality coefficient between the pressure and the displacement normal to the surface, and

$1/c_1$ is the proportionality coefficient between the pressure and the velocity normal to the surface.

This model can be conceptualized as a spring and dashpot in series placed between the acoustic medium and a rigid wall. The spring and dashpot parameters are k_1 and c_1, respectively, defined per unit area of the interface surface. These reactive acoustic boundaries can have a significant effect on the pressure distribution in the acoustic medium, in particular if the coefficients k_1 and c_1 are chosen such that the boundary is energy absorbing. If no impedance, loads, or fluid-solid coupling are specified on the surface of an acoustic mesh, the acceleration of that surface is assumed to be zero. This is equivalent to the presence of a rigid wall at that boundary.

Use of the subspace-based steady-state dynamics procedure is not recommended if reactive acoustic boundaries with strong absorption characteristics are used. Since the effect of c_1 is not taken into account in an eigenfrequency extraction step, the eigenmodes may have shapes that are significantly different from the exact solution.

Sloshing of a free surface

To model small-amplitude "sloshing" of a free surface in a gravity field, set $1/k_1 = 1/(\rho_f\, g)$ and $1/c_1 = 0$, where ρ_f is the density of the fluid and g is the gravitational acceleration (assumed to be directed normal to the surface). This relation holds for small volumetric drag.

Acoustic-structural interface

The impedance boundary condition can also be placed at an acoustic-structural interface. In this case the boundary condition can be conceptualized as a spring and dashpot in series placed between the acoustic medium and the structure. The expression for the outward velocity still holds, with $\dot{u}_{out}$ now being the relative outward velocity of the acoustic medium and the structure:

$$\dot{u}_{out} = \mathbf{n} \cdot (\dot{\mathbf{u}}^f - \dot{\mathbf{u}}^m),$$

where $\dot{\mathbf{u}}^m$ is the velocity of the structure, $\dot{\mathbf{u}}^f$ is the velocity of the acoustic medium at the boundary, and $\mathbf{n}$ is the outward normal to the acoustic medium.

Steady-state dynamics

In a steady-state dynamics analysis the expression for the outward velocity can be written in complex form as

$$\dot{u}_{out} = (\frac{1}{c_1} + i\,\frac{\Omega}{k_1})p = \frac{1}{Z(\Omega)}p,$$

where Ω is the circular frequency (radians/second) and we define

$$\frac{1}{Z(\Omega)} \equiv \frac{1}{c_1} + i\,\frac{\Omega}{k_1}.$$

The term $1/Z(\Omega)$ is the complex admittance of the boundary, and $Z(\Omega)$ is its complex impedance. Thus, a required complex impedance or admittance value can be entered for a given frequency by specifying the parameters $1/c_1$ and $1/k_1$.

Specifying impedance conditions

You specify impedance coefficient data in an impedance property table. You can describe an impedance table in terms of the admittance parameters, $1/c_1$ and $1/k_1$, or in terms of the real and imaginary parts of the impedance. In the latter case Abaqus converts the user-defined table of impedance data to the admittance parameter form for the analysis.

The parameters in the table can be specified over a range of frequencies. The required values are interpolated from the table in steady-state harmonic response analysis only; for other analysis types, only the first table entry is used. The name of the impedance property table is referred to from a surface-based or element-based impedance definition. In Abaqus/CAE impedance conditions are always surface-based; surfaces can be defined as collections of geometric faces and edges or collections of element faces and edges.

In a steady-state dynamics analysis you cannot specify impedance conditions on a surface on which incident wave loading is applied.

Input File Usage: Use the following option to specify an impedance using a table of admittance parameters (default):

∗IMPEDANCE PROPERTY, NAME=*impedance property table name*, DATA=ADMITTANCE

Use the following option to specify an impedance using a table of the real and imaginary parts of the impedance:

∗IMPEDANCE PROPERTY, NAME=*impedance property table name*, DATA=IMPEDANCE

Abaqus/CAE Usage: Use the following input to specify an impedance using a table of admittance parameters:

Interaction module: **Create Interaction Property**: **Name**: *impedance property table name* and **Acoustic impedance**: **Data type**: **Admittance**

Use the following input to specify an impedance using a table of the real and imaginary parts of the impedance:

Interaction module: **Create Interaction Property**: **Name**: *impedance property table name* and **Acoustic impedance**: **Data type**: **Impedance**

Specifying surface-based impedance conditions

You can define the impedance condition on a surface. The impedance is applied to element edges in two dimensions and to element faces in three dimensions. The element-based surface (see "Element-based surface definition," Section 2.3.2) contains the element and face information.

Input File Usage: ∗SIMPEDANCE, PROPERTY=*impedance property table name*
surface name

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Acoustic impedance**: select surface: **Definition**: **Tabular**, **Acoustic impedance property**: *impedance property table name*

Specifying element-based impedance conditions

Alternatively, you can define the impedance condition on element faces. The impedance is applied to element edges in two dimensions and to element faces in three dimensions. The edge or face of the element upon which the impedance is placed is identified by an impedance load type and depends on the element type (see Part VI, "Elements").

Input File Usage: ∗IMPEDANCE, PROPERTY=*impedance property table name*
element number or set name, *impedance load type label*

Abaqus/CAE Usage: Element-based impedance conditions are not supported in Abaqus/CAE. However, similar functionality is available using surface-based impedance conditions.

Modifying or removing impedance conditions

Impedance conditions can be added, modified, or removed as described in "Applying loads: overview," Section 32.4.1.

Radiation boundaries for exterior problems

An exterior problem such as a structure vibrating in an acoustic medium of infinite extent is often of interest. Such a problem can be modeled by using acoustic elements to model the region between the structure and a simple geometric surface (located away from the structure) and applying a radiating (nonreflecting) boundary condition at that surface. The radiating boundary conditions are approximate, so the error in an exterior acoustic analysis is controlled not only by the usual finite element discretization error but also by the error in the approximate radiation condition. In Abaqus the radiation boundary conditions converge to the exact condition in the limit as they become infinitely distant from the radiating structure. In practice, these radiation conditions provide accurate results when the surface is at least one-half wavelength away from the structure at the lowest frequency of interest.

Except in the case of a plane wave absorbing condition with zero volumetric drag, the impedance parameters in Abaqus/Standard are frequency dependent. The frequency-dependent parameters are used in the direct-solution and subspace-based steady-state dynamics procedures. In direct time integration procedures the zero-drag values for the constants $1/c_1$ and $1/k_1$ are used. These values will give good results when the drag is small. (Small volumetric drag here means $\gamma << \rho_f \Omega$, where ρ_f is the density of the acoustic medium and Ω is the circular excitation frequency or sound wave frequency.)

A direct-solution steady-state dynamics procedure ("Direct-solution steady-state dynamic analysis," Section 6.3.4) must include both real and complex terms if nonreflecting (also called quiet) boundaries are present, because nonreflecting boundaries represent a form of damping in the system. The use of the subspace-based steady-state dynamics procedure is not recommended if quiet boundaries are used.

Several radiating boundary conditions are implemented as special cases of the impedance boundary condition. The details of the formulation are given in "Coupled acoustic-structural medium analysis," Section 2.9.1 of the Abaqus Theory Manual.

Element-based impedance conditions are not supported in Abaqus/CAE. However, similar functionality is available using surface-based impedance conditions.

Planar nonreflecting boundary condition

The simplest nonreflecting boundary condition available in Abaqus assumes that the plane waves are normally incident on the exterior surface. This planar boundary condition ignores the curvature of the boundary and the possibility that waves in the simulation may impinge on the boundary at an arbitrary angle. The planar nonreflecting condition provides an approximation: acoustic waves are transmitted across such a boundary with little reflection of energy back into the acoustic medium. The amount of energy reflected is small if the boundary is far away from major acoustic disturbances and is reasonably orthogonal to the direction of dominant wave propagation. Thus, if an exterior (unbounded domain) problem is to be solved, the nonreflecting boundary should be placed far enough away from the sound source so that the assumption of normally impinging waves is sufficiently accurate. This condition would be used, for example, on the exhaust end of a muffler.

Input File Usage: Use either of the following options (default):

```
*SIMPEDANCE, NONREFLECTING=PLANAR
*IMPEDANCE, NONREFLECTING=PLANAR
```

Abaqus/CAE Usage: Use the following input to specify a surface-based planar nonreflecting boundary condition:

Interaction module: **Create Interaction**: **Acoustic impedance**: select surface: **Definition**: **Nonreflecting**, **Nonreflecting type**: **Planar**

Improved nonreflecting boundary condition for plane waves

For the planar nonreflecting boundary condition to be accurate, the plane waves must be normally incident to a planar boundary. However, the angle of incidence is generally unknown in advance. A radiating boundary condition that is exact for plane waves with arbitrary angles of incidence is available in Abaqus. The radiating boundary can have any arbitrary shape. This boundary impedance is implemented only for transient dynamics.

Input File Usage: Use either of the following options:

*SIMPEDANCE, NONREFLECTING=IMPROVED
*IMPEDANCE, NONREFLECTING=IMPROVED

Abaqus/CAE Usage: Use the following input to specify a surface-based improved planar nonreflecting boundary condition:

Interaction module: **Create Interaction**: **Acoustic impedance**: select surface: **Definition**: **Nonreflecting**, **Nonreflecting type**: **Improved planar**

Geometry-based nonreflecting boundary conditions

Four other types of absorbing boundary conditions that take the geometry of the radiating boundary into account are implemented in Abaqus: circular, spherical, elliptical, and prolate spheroidal. These boundary conditions offer improved performance over the planar nonreflecting condition if the nonreflecting surface has a simple, convex shape and is close to the acoustic sources. The various types of absorbing boundaries are selected by defining the required geometric parameters for the element-based or surface-based impedance definition.

The geometric parameters affect the nonreflecting surface impedance. To specify a nonreflecting boundary that is circular in two dimensions or a right circular cylinder in three dimensions, you must specify the radius of the circle. To specify a nonreflecting spherical boundary condition, you must specify the radius of the sphere. To specify a nonreflecting boundary that is elliptical in two dimensions or a right elliptical cylinder in three dimensions or to specify a prolate spheroid boundary condition, you must specify the shape, location, and orientation of the radiating surface. The two parameters specifying the shape of the surface are the semimajor axis and the eccentricity. The semimajor axis, a, of an ellipse or prolate spheroid is analogous to the radius of a sphere: it is one-half the length of the longest line segment connecting two points on the surface. The semiminor axis, b, is one-half the length of the longest line segment that connects two points on the surface and is orthogonal to the semimajor axis line. The eccentricity, ϵ, is defined as $\epsilon = \sqrt{1 - (b/a)^2}$.

See "Acoustic radiation impedance of a sphere in breathing mode," Section 1.11.3 of the Abaqus Benchmarks Manual, and "Acoustic-structural interaction in an infinite acoustic medium," Section 1.11.4 of the Abaqus Benchmarks Manual, for benchmark problems showing the use of these conditions.

Input File Usage: Use one of the following options:

*SIMPEDANCE, NONREFLECTING=CIRCULAR
*SIMPEDANCE, NONREFLECTING=SPHERICAL
*SIMPEDANCE, NONREFLECTING=ELLIPTICAL
*SIMPEDANCE, NONREFLECTING=PROLATE SPHEROIDAL

In each case, the *IMPEDANCE element-based option can be used instead of *SIMPEDANCE.

Abaqus/CAE Usage: Use the following input to specify surface-based geometric nonreflecting boundary conditions:

Interaction module: **Create Interaction**: **Acoustic impedance**: select surface: **Definition**: **Nonreflecting**, **Nonreflecting type**: **Circular**, **Spherical**, **Elliptical**, or **Prolate spheroidal**

Combining different radiation conditions in the same problem

Since the radiation boundary conditions for the different shapes are spatially local and do not involve discretization in the infinite exterior domain, an exterior boundary can consist of the combination of several shapes. The appropriate boundary condition can then be applied to each part of the boundary. For example, a circular cylinder can be terminated with hemispheres (see "Fully and sequentially coupled acoustic-structural analysis of a muffler," Section 9.1.1 of the Abaqus Example Problems Manual), or an elliptical cylinder can be terminated with prolate spheroidal halves. This modeling technique is most effective if the boundaries between surfaces are continuous in slope as well as displacement, although this is not essential.

Concentrated pressure-conjugate load

Distributed "loads" on acoustic elements can be interpreted as normal pressure gradients per unit density (dimensions of force per unit mass or acceleration). When used in Abaqus, the applied distributed loads must be integrated over a surface area, yielding a quantity with dimensions of force times area per unit mass (or volumetric acceleration). For analyses in the frequency domain and for transient dynamic analyses where the volumetric drag is zero, this acoustic load is equal to the volumetric acceleration of the fluid on the boundary. For example, a horizontal, flat rigid plate oscillating vertically imposes an acceleration on the acoustic fluid and an acoustic "load" equal to this acceleration times the surface area of the plate. For the transient dynamics formulation in the presence of volumetric drag, however, the specified "load" is slightly different. It is also a force times area per unit mass; but this force effect is partially lost to the volumetric drag, so the resulting volumetric acceleration of the fluid on the boundary is reduced. Noting this distinction for the special case of volumetric drag and transient dynamics, it is nevertheless convenient to refer to acoustic "loads" as volumetric accelerations in general.

An inward volumetric acceleration can be applied by a positive concentrated load on degree of freedom 8 at a node of an acoustic element that is on the boundary of the acoustic medium. In Abaqus/Standard you can specify the in-phase (real) part of a load (default) and the out-of-phase (imaginary) part of a load. Inward particle accelerations (force per unit mass in transient dynamics) on

the face of an acoustic element should be lumped to concentrated loads representing inward volumetric accelerations on the nodes of the face in the same way that pressure on a face is lumped to nodal forces on stress/displacement elements.

Input File Usage: Use the following option to define the real part of the load:

*CLOAD, REAL

Use the following option to define the imaginary part of the load:

*CLOAD, IMAGINARY

Abaqus/CAE Usage: Load module: **Create Load**: choose **Acoustic** for the **Category** and **Inward volume acceleration** for the **Types for Selected Step**

Incident wave loading due to external sources

Abaqus provides a type of distributed load for loads due to external wave sources. Individual spherical monopole or individual or diffuse planar sources can be defined, subjecting the fluid and solid region of interest to an incident field of waves. Waves produced by an explosion or sound source propagate from the source, impinging on and passing over the structure, producing a temporally and spatially varying load on the structural surface. In the fluid the pressure field is affected by reflections and emissions from the structure as well as by the incident field from the source itself. The incident wave loads on acoustic and/or solid meshes depend on the location of the source node, the properties of the propagating fluid, and the reference time history or frequency dependence specified at the reference ("standoff") node as indicated in Figure 32.4.6–1.

Several distinct modeling methods can be used in Abaqus with incident wave loading, requiring different approaches to applying the incident wave loads. For problems involving solid and structural elements only (for example, where the incident wave field is due to waves in air) the wave loading is applied roughly like a distributed surface load. This might apply to an analysis of blast loads in air on a vehicle or building (see "Example: airblast loading on a structure," shown in Figure 32.4.6–6). In Abaqus/Explicit the CONWEP model can be used for air blast loading on solid and structural elements, without the need to model the fluid medium. "Deformation of a sandwich plate under CONWEP blast loading," Section 9.1.8 of the Abaqus Example Problems Manual, is an example of a blast loading problem.

Incident wave loads (with the exception of CONWEP loading) can be applied to beam structures as well; this is a common modeling method for ship whipping analysis and for steel frame buildings subject to blast loads. Incident wave loads can be applied to surfaces defined on two- or three-dimensional beam elements. However, incident wave loads can be applied only to three-dimensional beams for transient dynamic analysis where beam fluid inertia is defined. Incident wave loads cannot be defined on frame elements, line spring elements, three-dimensional open-section beam elements, or three-dimensional Euler-Bernoulli beams.

In underwater explosion analyses (for example, a ship or submerged vehicle subjected to an underwater explosion loading as depicted in Figure 32.4.6–4 and Figure 32.4.6–5) the fluid is also discretized using a finite element model to capture the effects of the fluid stiffness and inertia. For these problems involving both solid and acoustic elements, two formulations of the acoustic pressure field

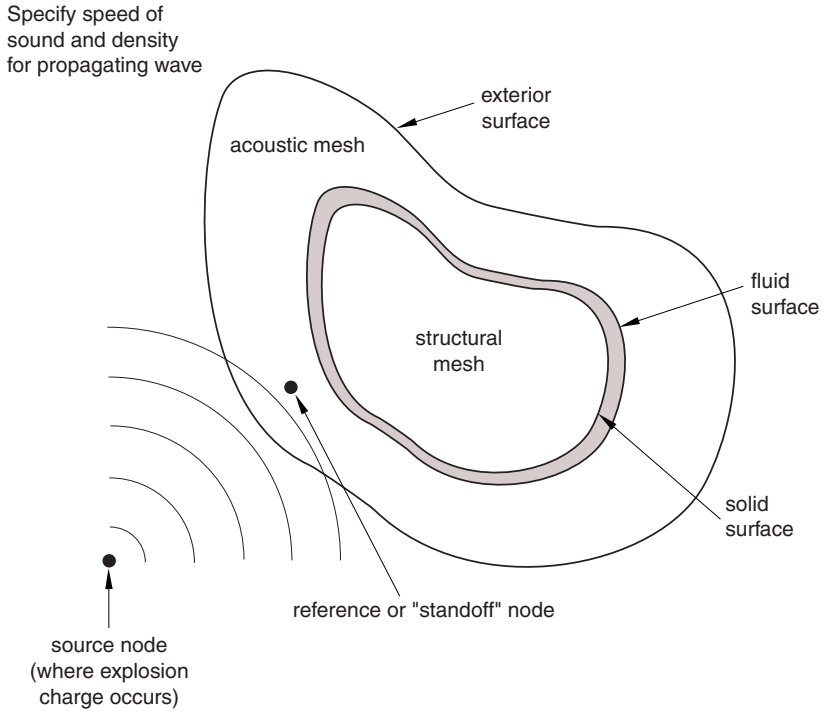


Figure 32.4.6–1 Incident wave loading model.

exist. First, the acoustic elements can be used to model the total pressure in the medium, including the effects of the incident field and the overall system's response. Alternatively, the acoustic elements can be used to model only the response of the medium to the wave loads, not the wave pulse itself. The former case will be referred to as the "total wave" formulation, the latter as the "scattered wave" formulation.

Incident wave interactions are also used to model sound fields impinging on structures or acoustic domains. The acoustic field scattered by a structure or the sound transmitted through the structure may be of interest. Usually, sound scattering and transmission problems are modeled using the scattered formulation with steady-state dynamic procedures. Transient procedures can also be used, in a manner analogous to underwater explosion analysis problems.

Scattered and total wave formulations

The distinction between the total wave formulation and the scattered wave formulation is relevant only when incident wave loads are applied. The total wave formulation is more closely analogous to structural loading than the scattered wave formulation: the boundary of the acoustic medium is specified as a loaded surface, and a time-varying load is applied there, which generates a response in the acoustic medium. This response is equal to the total acoustic pressure in the medium. The scattered wave formulation

exploits the fact that when the acoustic medium is linear, the response in the medium can be decomposed into a sum of the incident wave and the scattered field. The total wave formulation must be used when the acoustic medium is nonlinear due to possible fluid cavitation (see "Loading due to an incident dilatational wave field," Section 6.3.1 of the Abaqus Theory Manual).

Table 32.4.6–1 describes the procedure types for which each formulation is supported.

Table 32.4.6–1 Supported procedures for scattered and total wave formulations.

Procedure	Scattered	Total Wave
Steady-state dynamics	Yes	No
Transient	Yes	Yes

Scattered wave formulation

When the mechanics of a fluid can be described as linear, the observed total acoustic pressure can be decomposed into two components: the known incident wave and the "scattered" wave that is produced by the interaction of the incident wave with structures and/or fluid boundaries. When this superposition is applicable, it is common practice to seek the "scattered" wave field solution directly. When using the scattered wave formulation, the pressures at the acoustic nodes are defined to be only the scattered part of the total pressure. Both acoustic and solid surfaces at the acoustic-structural interface should be loaded in this case.

When using incident wave loads in steady-state dynamic procedures, the scattered wave formulation must be used.

Input File Usage: Use the following option to specify the scattered wave formulation (default):

```
*ACOUSTIC WAVE FORMULATION, TYPE=SCATTERED WAVE
```

Abaqus/CAE Usage: Any module: **Model→Edit Attributes→*model_name***. Toggle on **Specify acoustic wave formulation**: select **Scattered wave**

Total wave formulation

The total wave formulation (see "Coupled acoustic-structural medium analysis," Section 2.9.1 of the Abaqus Theory Manual) is particularly applicable when the acoustic medium is capable of cavitation, rendering the fluid mechanical behavior nonlinear. It should also be used if the problem contains either a curved or a finite extent boundary where the pressure history is prescribed. Only the outer acoustic surfaces should be loaded with the incident wave in this case, and the incident wave source must be located exterior to the fluid model. Any impedance or nonreflecting condition that may exist on this outer acoustic boundary applies only on the part of the acoustic solution that does not include the prescribed incident wave field (that is, only the scattered field is subject to the nonreflecting condition). Thus, the applied incident wave loading will travel into the problem domain without being affected by the nonreflecting conditions on the outer acoustic surface.

In the total wave formulation the acoustic pressure degree of freedom stands for the total dynamic acoustic pressure, including contributions from incident and scattered waves and, in Abaqus/Explicit, the

dynamic effects of fluid cavitation. The pressure degree of freedom does not include the acoustic static pressure, which can be specified as an initial condition (see "Defining initial acoustic static pressure" in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1). This acoustic static pressure is used only in determining the cavitation status of the acoustic element nodes and does not apply any static loads to the acoustic or structural mesh at their common wetted interface. It does not apply to analyses using Abaqus/Standard.

Input File Usage: Use the following option to specify the total wave formulation:

*ACOUSTIC WAVE FORMULATION, TYPE=TOTAL WAVE

Abaqus/CAE Usage: Any module: **Model→Edit Attributes→*model_name***. Toggle on **Specify acoustic wave formulation**: select **Total wave**

Initialization of acoustic fields

For transient dynamics, when the total wave formulation is used with the incident wave standoff point located inside the acoustic finite element domain, the acoustic solution is initialized to the values of the incoming incident wave. This initialization is performed automatically, for pressure-based incident wave amplitude definitions only, at the beginning of the first direct-integration dynamic step in an analysis; in restarted analyses, steps are counted from the beginning of the initial analysis. This initialization not only saves computational time but also applies the incident wave loading without significant numerical dissipation or distortion. During the initialization phase all incident wave loading definitions in the first dynamic analysis step are considered, and all acoustic element nodes are initialized to the incident wave field at time zero. Incident wave loads specified with different source locations count as separate load definitions for the purpose of initialization of the acoustic nodes. Any reflections of the incident wave loads are also taken into account during the initialization phase.

Describing incident wave loading

To use incident wave loading, you must define the following:

- information that establishes the direction and other properties of the incident wave,
- the time history or frequency dependence of the source pulse at some reference ("standoff") point,
- the fluid and/or solid surfaces to be loaded, and
- any reflection plane outside the problem domain, such as a seabed in an underwater explosion study, that would reflect the incident wave onto the problem domain.

Two interfaces are available in Abaqus for applying incident wave loads: a preferred interface that is supported in Abaqus/CAE and an alternative interface that has been available in previous releases and is not supported in Abaqus/CAE. The preferred interface is conceptually the same as the alternative interface and uses essentially the same data. The preferred interface options include the term "interaction" to distinguish them from the incident wave and incident wave property options of the alternative interface. Unless otherwise specified, the discussion in this section applies to both of the interfaces. The usages for the preferred interface are included in the discussion; the usages for the alternative interface are described in "Alternative incident wave loading interface," below. Refer to the

example problems discussed at the end of this section to see how the incident wave loading is specified using the preferred interface.

Prescribing geometric properties and the speed of the incident wave

You must refer to a property definition for each prescribed incident wave. Incident wave loads in Abaqus may be either planar, spherical, or diffuse. You select a planar incident wave (default), spherical incident wave, or a diffuse field in the incident wave property definition.

Planar incident waves maintain constant amplitude as they travel in space; consequently, the speed and direction of travel are the critical parameters to define. The speed is defined in the incident wave interaction property definition, and the direction is determined by the locations of the source and standoff points you define as part of the incident wave interaction.

For spherical incident wave definitions, the wave reduces in amplitude as a function of space. By default, the amplitude of a spherical wave is inversely proportional to the distance from the source; this behavior is called "acoustic" propagation. For the preferred interface you can modify the default propagation behavior to define spatial decay of the incident wave field. The dimensionless constants A, B, and C are used to define the spatial decay as a function of the distance R_j between the source point and the loaded point and the distance R_0 between the source point and the standoff point:

$$p_x(R_0, R_j) \equiv \left(\frac{R_0}{R_j}\right)^{\left[\frac{(A+1)R_j}{CR_0+(B+1)R_j}\right]}.$$

Refer to "Loading due to an incident dilatational wave field," Section 6.3.1 of the Abaqus Theory Manual, for details of the generalized spatial decay formulation.

In Abaqus incident wave interactions can be used to simulate diffuse incident fields. Diffuse fields are characteristic of reverberant spaces or other situations in which waves from many directions strike a surface. For example, reverberant chambers are constructed intentionally in acoustic test facilities for sound transmission loss measurements. The diffuse field model used in Abaqus, as shown in Figure 32.4.6–2, allows you to specify a seed number N; N^2 deterministic incident plane waves travel along vectors distributed over a hemisphere so that the incident power per solid angle approximates a diffuse incident field.

The fluid and the solid surfaces where the incident loading acts are specified in the incident wave loading definition. The incoming wave load is further described by the locations of its source point and of a reference ("standoff") point where the wave amplitude is specified. For information on how to specify these surfaces and the standoff point, see "Identifying the fluid and the solid surfaces for incident wave loading," and "The standoff point" below. For a planar wave the specified locations of the source and the standoff points are used to define the direction of wave propagation.

The speed of the incident wave is prescribed by giving the properties for the incident wave-bearing acoustic medium. These specified properties should be consistent with the properties specified for the fluid discretized using acoustic elements.

For the preferred interface you must define nodes corresponding to the source and standoff points for the incident wave; the node numbers or set names must be specified for each incident wave definition.

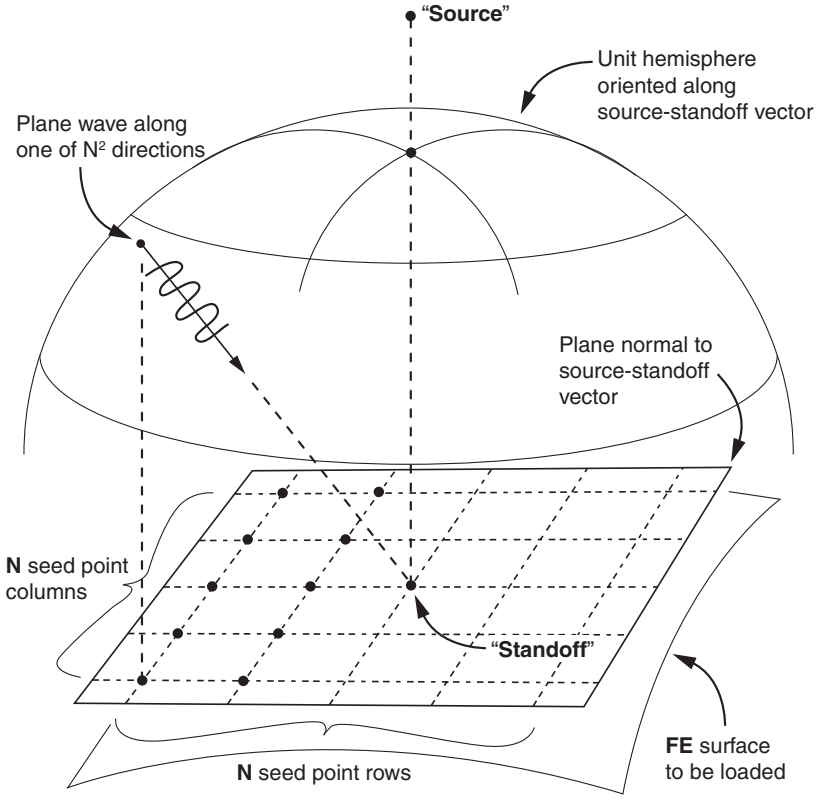


Figure 32.4.6–2 Diffuse loading model.

The node set names, if used, must contain only a single node. Neither the source node nor the standoff node should be connected to any elements in the model.

Input File Usage:

∗INCIDENT WAVE INTERACTION PROPERTY,
NAME=*wave property name*, TYPE=PLANE or SPHERE
speed of sound, fluid mass density, A, B, C
∗INCIDENT WAVE INTERACTION, PROPERTY=*wave property name*
fluid surface name, *source node*, *standoff node*, *reference magnitude*

The constants *A*, *B*, and *C* apply only for spherical incident waves with generalized spatial decay propagation.

∗INCIDENT WAVE INTERACTION PROPERTY,
NAME=*wave property name*, TYPE=DIFFUSE
speed of sound, fluid mass density
∗INCIDENT WAVE INTERACTION, PROPERTY=*wave property name*
fluid surface name, *source node*, *standoff node*, *reference magnitude, N*

The seed number *N* generates planar incident waves with directions distributed on a hemisphere centered at the standoff point.

Abaqus/CAE Usage: Interaction module: **Create Interaction Property**: **Name**: *wave property name* and **Incident wave**, **Speed of sound in fluid**: *speed of sound*, **Fluid density**: *fluid mass density*

Select one of the following definitions:

Definition: **Planar**
Definition: **Spherical**, **Propagation model**: **Acoustic**
Definition: **Spherical**, **Propagation model**: **Generalized decay**, enter values for **A**, **B**, and **C**
Definition: **Diffuse**, **Seed number**: *N*

Create Interaction: **Incident wave**: select the source point, select the standoff point, select the region: **Wave property**: *wave property name*, **Reference magnitude**: *reference magnitude*

Identifying the fluid and the solid surfaces for incident wave loading

In the scattered wave formulation the incident wave loading must be specified on all fluid and solid surfaces that reflect the incident wave with two exceptions:

- those fluid surfaces that have the pressure values directly prescribed using boundary conditions; and
- those fluid surfaces that have symmetry conditions (the symmetry must hold for both the loading and the geometry).

In problems with a fluid-solid interface both surfaces must be specified in the incident wave loading definition for the scattered formulation. See "Example: submarine close to the free surface," shown in Figure 32.4.6–4.

When the total pressure-based formulation is specified, the incident wave loading must be specified only on the fluid surfaces that border the infinite region that is excluded from the model. Typically, these surfaces have a nonreflecting radiation condition specified on them, and the implementation ensures that the radiation condition is enforced only on the scattered response of the modeled domain and not on the incident wave itself. See "Example: submarine close to the free surface," and "Example: surface ship," shown in Figure 32.4.6–4 and Figure 32.4.6–5, respectively.

In certain problems, such as blast loads in air, you may decide that the blast wave loads on a structure need to be modeled, but the surrounding fluid medium itself does not. In these problems the incident wave loading is specified only on the solid surfaces since the fluid medium is not modeled. The distinction between the scattered wave formulation and the total wave formulation for handling the incident wave loading is not relevant in these problems since the wave propagation in the fluid medium is of no interest.

The standoff point

In transient analyses the standoff point is a reference point used to specify the pulse loading time history: it is the point at which the user-defined pulse history is assumed to apply with no time delay, phase shift,

or spreading loss. In steady-state analyses using discrete planar or spherical sources, the standoff point is the point at which the incident field has zero phase.

In transient analyses the standoff point should be defined so that it is closer to the source than any point on the surfaces in the model that would reflect the incident wave. Doing so ensures that all the points on these surfaces will be loaded with the specified time history of the source and that the analysis begins before the wave overtakes any portion of these surfaces. To save analysis time, the standoff point is typically on or near the solid surface where the incoming incident wave would be first deflected (see "Example: submarine close to the free surface," shown in Figure 32.4.6–4). However, the standoff point is a fixed point in the analysis: if the loaded surfaces move before the incident wave loading begins, due to previous analysis steps or geometric adjustments, the surfaces may envelop the specified standoff point. Care should be taken to define a standoff point such that it remains closer to the incident wave source point than any point on the loaded surfaces at the onset of the loading.

When the total wave formulation is used and the incident wave loading is specified in the first step of the analysis in terms of pressure history, Abaqus automatically initializes the pressure and the pressure rate at the acoustic nodes to values based on the incident wave loading. This allows the acoustic analysis to start with the incident waves partially propagated into the problem domain at time zero and assumes that this propagation had taken place with negligible effect of any volumetric dissipative sources such as the fluid drag. When the incident wave loading is specified in terms of the pressure values, the recommendations given above for selecting a standoff point are valid with the total wave formulation as well. However, when the incident wave loading is specified in terms of acceleration values, the automatic initialization is not done and the standoff point should be located near the exterior fluid boundary of the model such that the standoff point is closer to the source than any point on the exterior boundary. See "Example: submarine close to the free surface," and "Example: surface ship," shown in Figure 32.4.6–4 and Figure 32.4.6–5, respectively.

In steady-state analyses the role of the standoff point is somewhat different. When the incident wave interaction property is of planar or spherical type, you define the real and imaginary parts of the magnitude at the standoff point. Separately, the specified real and imaginary incident waves are taken to have zero phase at the standoff point (combined, these two waves could be equivalent to a single wave with nonzero phase at the standoff). Every location on the loaded surface has a phase shift in the applied pressure or acoustic traction, corresponding to the difference in propagation time between the loaded point and the standoff. This means that an incident wave defined, for example, with a pure real value at the standoff point generates both real and imaginary tractions at all the other points on the loaded surface.

When the incident wave is of diffuse type, the role of the standoff and source points is primarily to orient the loaded surface with respect to the incoming reverberant field. The model used for diffuse incident wave loading applies a set of deterministically defined plane waves, whose directions are defined as vectors connecting the standoff point and an array of points on a hemisphere. This hemisphere is centered at the standoff point, and its apex is the source point. The array of points is set according to the specified seed, N, and a deterministic algorithm that arranges N^2 points on the hemisphere. The algorithm concentrates the points so that the incident waves in the diffuse field model are concentrated at normal incidence, with fewer waves at oblique angles. The specified amplitude value and reference magnitude are divided equally among the N^2 incident waves. The orientation of

the hemisphere containing the incident waves in the diffuse model is the same for all of the points on the loaded surface—it does not vary with the local normal vector on the surface.

Defining the amplitude of the source pulse

For transient analyses the time history to be specified by the user is that observed at the standoff point: histories at a point on the loaded surface are computed from the wave type and the location of that point relative to the standoff point. The time history of the acoustic source pulse can be defined either in terms of the fluid pressure values or the fluid particle acceleration values. Pressure time histories can be used for any type of element, such as acoustic, structural, or solid elements; acceleration time histories are applicable only for acoustic elements. In either case a reference magnitude is specified for any given incident-wave-loaded surface, and a reference to a time-history data table defined by an amplitude curve is specified. The reference magnitude varies with time according to the amplitude definition.

For steady-state dynamic analyses the amplitude definition specified as part of the incident wave interaction definition is interpreted as the frequency dependence of the wave at the standoff point.

Currently the source pulse description in terms of fluid particle acceleration history is limited to planar incident waves acting on fluid surfaces in transient analyses. Further, if an impedance condition is specified on the same fluid surface along with incident wave loading, the source pulse is restricted to the pressure history type even for planar incident waves. The source pulse in terms of pressure history can be used without these limitations; i.e., pressure-history-based incident wave loading can be used with fluid or solid surfaces, with or without impedance, and for both planar and spherical incident waves.

When the source pulse is specified using pressure values and is applied on a fluid surface, the pressure gradient is computed and applied as a pressure-conjugate load on these surfaces. Hence, it is desirable to define the pulse amplitude to begin with a zero value, particularly when the cavitation in the fluid is a concern. If the structural response is of primary concern and the scattered formulation is being used, any initial jump in the pressure amplitude can be addressed by applying additional concentrated loads on the structural nodes that are tied to the acoustic mesh, corresponding to the initial jump in the incident wave pressure amplitude. Clearly, the additional load on any given structural node should be active from the instance the incident wave first arrives at that structural node. However, the scattered wave solution in the fluid still needs careful interpretation taking the initial jump into account.

Input File Usage: Use the following option to define the time history in terms of fluid pressure values:

∗INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=*amplitude data table name*
solid or fluid surface name, *source node*, *standoff node*, *reference magnitude*

Use the following option to define the time history in terms of fluid particle acceleration values:

∗INCIDENT WAVE INTERACTION, ACCELERATION
AMPLITUDE=*amplitude data table name*
fluid surface name, *source node*, *standoff node*, *reference magnitude*

Use the following option to define the real part of the loading (default):

*INCIDENT WAVE INTERACTION, REAL

Use the following option to define the imaginary part of the loading:

*INCIDENT WAVE INTERACTION, IMAGINARY

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Incident wave**: select the source point, select the standoff point, select the region: **Reference magnitude**: *reference magnitude*

Use the following options to define the time history in terms of fluid pressure values or fluid particle acceleration values:

Definition: **Pressure** or **Acceleration**, **Pressure amplitude** or **Acceleration amplitude**: *amplitude data table name*

Use the following options to define the real or imaginary part of the loading:

Toggle on **Real amplitude** and/or **Imaginary amplitude**: *amplitude data table name*

Defining bubble loading for spherical incident wave loading

An underwater explosion forms a highly compressed gas bubble that interacts with the surrounding water, generating an outward-propagating shock wave. The gas bubble floats upward as it generates these waves changing the relative positions of the source and the loaded surfaces. The loading effects due to bubble formation can be defined for spherical incident wave loading by using a bubble definition in conjunction with the incident wave loading definition.

The bubble dynamics can be described using a model internal to Abaqus or by using tabulated data. Abaqus has a built-in mechanical model of the bubble interacting with the surrounding fluid, which is simulated numerically to generate a set of data prior to running the finite element analysis. You can specify the explosive material parameters, ending time, and other parameters that affect the computation of the bubble amplitude curve used, as shown in Table 32.4.6–2.

Table 32.4.6–2 Parameters that define the bubble behavior.

Name	Dimensions	Description	Default
K	$\mathrm{FL}^{-2}(\mathrm{LM}^{-1/3})^{1+\mathrm{A}}$	Charge constant	None
k	$\mathrm{T}/(\mathrm{M}^{\frac{1-\mathrm{B}}{3}}\mathrm{L}^{\mathrm{B}})$	Charge constant	None
A	Dimensionless	Similitude spatial exponent	None
B	Dimensionless	Similitude temporal exponent	None
K_c	F/L^2	Charge constant	None
γ	Dimensionless	Ratio of specific heats for explosion gas	None

Name	Dimensions	Description	Default
ρ_c	M/L^3	Charge material density	None
m_c	M	Mass of charge	None
d_I	L	Initial charge depth	None
$\mathbf{n}_X$	Dimensionless	*X*-direction cosine of the free surface normal	None
$\mathbf{n}_Y$	Dimensionless	*Y*-direction cosine of the free surface normal	None
$\mathbf{n}_Z$	Dimensionless	*Z*-direction cosine of the free surface normal	None
g	L/T^2	Acceleration due to gravity	None
p_{atm}	F/L^2	Atmospheric pressure at free surface	None
η	Dimensionless	Wave effect parameter	1.0
C_D	Dimensionless	Bubble drag coefficient	0.0
E_D	Dimensionless	Bubble drag exponent	2.0
T_{final}	T	Maximum allowable time in bubble simulation	None
N_{steps}	Dimensionless	Maximum allowable number of steps in bubble simulation	1500
Ω_{rel}	Dimensionless	Relative error tolerance parameter for bubble simulation	1×10^{-11}
X_{abs}	Dimensionless	Absolute error tolerance parameter for bubble simulation	1×10^{-11}
β	Dimensionless	Error control exponent for bubble simulation	0.2
ρ_f	M/L^3	Fluid mass density	None
c_f	L/T	Fluid speed of sound	None

All of the parameters specified affect only the bubble amplitude; other physical parameters in the problem are independent. You can suppress the effects of wave loss in the bubble dynamics and introduce empirical flow drag, if desired. Detailed information about the bubble mechanical model is given in "Loading due to an incident dilatational wave field," Section 6.3.1 of the Abaqus Theory Manual.

In an underwater explosion event a bubble migrates upward toward, and possibly reaches, the free water surface. If the bubble migration reaches the free water surface during the specified analysis time, Abaqus applies loads of zero magnitude after this point.

Model data about the bubble simulation are written to the data (**.dat**) file. During an Abaqus/Standard analysis history data are written each increment to the output database (**.odb**) file. The history data include the radius of the bubble and the bubble depth below the free water surface. For reference, the pressure and acoustic load quantities at the standoff point are also written to the data file; these load terms include the direct plane-wave term and the spherical spreading ("afterflow") effect (see "Loading due to an incident dilatational wave field," Section 6.3.1 of the Abaqus Theory Manual).

For the preferred interface the loading effects due to bubble formation can be defined for spherical incident wave loading using the UNDEX charge property definition. Because the bubble simulation uses spherical symmetry, the incident wave interaction property must define a spherical wave.

You can also specify incident wave loading due to bubble dynamics using tabulated data for the pressure and source migration. For the preferred interface you specify independent amplitude curves for the pressure at the standoff point and any source node location time histories. The source location amplitude names are referred to from boundary condition definitions for the source node.

Input File Usage: Use the following options to specify loading effects due to bubble formation using the UNDEX charge property definition:

∗INCIDENT WAVE INTERACTION PROPERTY,
NAME=*wave property name*, TYPE=SPHERE
∗UNDEX CHARGE PROPERTY
data defining the UNDEX charge
∗INCIDENT WAVE INTERACTION, PROPERTY=*wave property name*,
UNDEX
fluid surface name, *source node*, *standoff node*, *reference magnitude*

Use the following options to specify pressure at the standoff point using tabulated data:

∗AMPLITUDE, DEFINITION=TABULAR, NAME=*pressure*
∗INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=*pressure*
solid or fluid surface name, *source node*, *standoff node*, *reference magnitude*

Use the following options to specify source node location time histories using tabulated data:

∗AMPLITUDE, DEFINITION=TABULAR, NAME=*name*
∗BOUNDARY, TYPE=DISPLACEMENT *or* VELOCITY,
AMPLITUDE=*name*
source node, *degrees of freedom*

Abaqus/CAE Usage: Use the following input to specify loading effects due to bubble formation using the UNDEX charge property definition:

Interaction module: **Create Interaction Property**: **Name**: *wave property name* and **Incident wave**: **Definition**: **Spherical**, **Propagation model**: **UNDEX charge**, enter data defining the UNDEX charge
Create Interaction: **Incident wave**: **Definition**: **UNDEX**, **Wave property**: *wave property name*, enter data defining the UNDEX charge

Use the following input to specify pressure at the standoff point using tabulated data:

Load or Interaction module: **Create Amplitude**: **Name**: *pressure* and select **Tabular**
Interaction module: **Create Interaction**: **Incident wave**: select the standoff point: **Definition**: **Pressure**, **Pressure amplitude**: *pressure*

Use the following input to specify source node location time histories using tabulated data:

Load or Interaction module: **Create Amplitude**: **Name**: *name* and select **Tabular**
Load module: **Create Boundary Condition**: select step: **Displacement/Rotation** or **Velocity/Angular velocity**: select the source node as the region and toggle on the degree or degrees of freedom, **Amplitude**: *name*

Modeling incident wave loading on a moving structure

To model the effect of relative motion between a structure (such as a ship) and the wave source during the analysis using the preferred interface, the source node may be assigned a velocity. It is assumed that the entire fluid-solid model is moving at a velocity with respect to the source node during the loading and that the speed of the model's motion is low compared to the speed of propagation of the incident wave. That is, the effect of the speed of the source is neglected in the computation of the loads, but the change in position of the source is included. This is equivalent to assuming that the relative motion between the source and the model is at a low Mach number. Relative motion can be specified only for transient analyses.

Input File Usage: Use the following option to assign a velocity to the source node:

*BOUNDARY, TYPE=DISPLACEMENT *or* VELOCITY, AMPLITUDE=*name*
source node, *degrees of freedom*

Abaqus/CAE Usage: Load module: **Create Boundary Condition**: select step: **Velocity/Angular velocity** or **Displacement/Rotation**: select regions and toggle on the degree or degrees of freedom, **Amplitude**: *name*

Specifying the reflection effects

The waves emanating from the source may reflect off plane surfaces, such as seabeds or sea surfaces, before reaching the specified standoff point. Thus, the incident wave loading consists of the waves arriving from a direct path from the source, as well as those arriving from reflections off the planes. In Abaqus an arbitrary number of these planes can be defined, each with its own location, orientation, and reflection coefficient.

If no reflection coefficient is specified, the plane is assumed to be nonreflective; a zero reflected pressure is applied. If a reflection coefficient is specified, the magnitude of the reflected waves are modified by the reflection coefficient Q according to the formula:

$$p_{reflected} = Qp_{incident}.$$

Only real values for Q are used.

The reflection planes are allowed only for incident waves that are defined in terms of fluid pressure values. Only one reflection off each plane is considered. If the effect of many successive reflections is important, these surfaces should be part of the finite element model. Reflection planes should not be used at a boundary of the finite element model if the total wave formulation is used, since in that case the incident wave will be reflected automatically by that boundary.

Input File Usage: Use the following option in conjunction with the *INCIDENT WAVE INTERACTION option to define an incident wave reflection plane:

*INCIDENT WAVE REFLECTION

Abaqus/CAE Usage: Incident wave reflections are not supported in Abaqus/CAE.

Boundary with prescribed pressure

The acoustic pressure degree of freedom at nodes of acoustic elements can be prescribed using a boundary condition. However, since you can use the nodal acoustic pressure in an Abaqus analysis to refer to the total pressure at that point or to only the scattered component, care must be exercised in some circumstances.

When the total wave formulation is used, a boundary condition alone is sufficient to specify a prescribed total dynamic pressure on a boundary.

In an analysis without incident wave loading, the nodal degree of freedom is generally equal to the total acoustic pressure at that point. Therefore, its value can be prescribed using a boundary condition in a manner consistent with other boundary conditions in Abaqus. For example, you may set the acoustic pressure at all of the nodes at a duct inlet to a prescribed amplitude to analyze the propagation of waves along the duct. The free surface of a body of water can be modeled by setting the acoustic pressure to zero at the surface.

When incident wave loading is used, the scattered wave formulation defines the nodal acoustic degree of freedom to be equal to the scattered pressure. Consequently, a boundary condition definition for this degree of freedom affects the scattered pressure only. The total acoustic pressure at a node is not directly accessible in this formulation. Specification of the total pressure in a scattered formulation

analysis is nevertheless required in some instances (for example, when modeling a free surface of a body of water). In this case, one of the following methods should be used.

If the fluid surface with prescribed total pressure is planar, unbroken, and of infinite extent, an incident wave reflection plane and a boundary condition can be used together to model the fact that the total pressure is zero on the free surface. A "soft" incident wave reflection plane coincident with the free surface will make sure that the structure is subjected to the incident wave load reflected off the free surface. A boundary condition setting the acoustic pressure in the surface equal to zero will make sure that any scattered waves emitted by the structure are reflected properly. The scattered wave solution in the fluid must be interpreted taking into consideration the fact that the incident field now includes a reflection of the source as well. If the fluid surface with prescribed total pressure is planar but broken by an object, such as a floating ship, this modeling technique may still be applied. However, the reflected loads due to the incident wave are computed as if the reflection plane passes through the hull of the ship; this approximation neglects some diffraction effects and may or may not be applicable in all situations of interest.

Alternatively, the free surface condition of the fluid can be eliminated by modeling the top layer of the fluid using structural elements, such as membrane elements, instead of acoustic elements. The "structural fluid" surface and the "acoustic fluid" surface are then coupled using either a surface-based mesh tie constraint ("Mesh tie constraints," Section 33.3.1) or, in Abaqus/Standard, acoustic-structural interface elements; and the incident wave loading must be applied on both the "structural fluid" and the "acoustic fluid" surfaces. The material properties of the "structural fluid" elements should be similar to those of the adjacent acoustic fluid. In Abaqus/Explicit the thickness of the "structural fluid" elements must be such that the masses at nodes on either side of the coupling constraint are nearly equal. This modeling technique allows the geometry of the surface on which total pressure is to be prescribed to depart from an unbroken, infinite plane. As a secondary benefit of this technique, you can obtain the velocity profile on the free surface since the displacement degrees of freedom are now activated at the "structural fluid" nodes. If a nonzero pressure boundary condition is desired, it can be applied as a distributed loading on the other side of the "structural fluid" elements.

Input File Usage: Use the following options for the first modeling technique with the default scattered wave formulation:

```
*BOUNDARY
*INCIDENT WAVE REFLECTION
```

Use the following option for the second modeling technique with the default scattered wave formulation:

```
*TIE
*INCIDENT WAVE INTERACTION
```

Use the following option with the total wave formulation:

```
*BOUNDARY
```

Abaqus/CAE Usage: Load module: **Create BC**: choose **Other** for the **Category** and **Acoustic pressure** for the **Types for Selected Step**

Defining air blast loading for incident shock waves using the CONWEP model in Abaqus/Explicit

An explosion in air forms a highly compressed gas mass that interacts with the surrounding air, generating an outward-propagating shock wave. The loading effects due to an explosion in air can be defined, for spherical incident waves (air blast) or hemispherical incident waves (surface blast), by empirical data provided by the CONWEP model in conjunction with the incident wave loading definition.

Unlike an acoustic wave, a blast wave corresponds to a shock wave with discontinuities in pressure, density, etc. across the wave front. Figure 32.4.6–3 shows a typical pressure history of a blast wave.

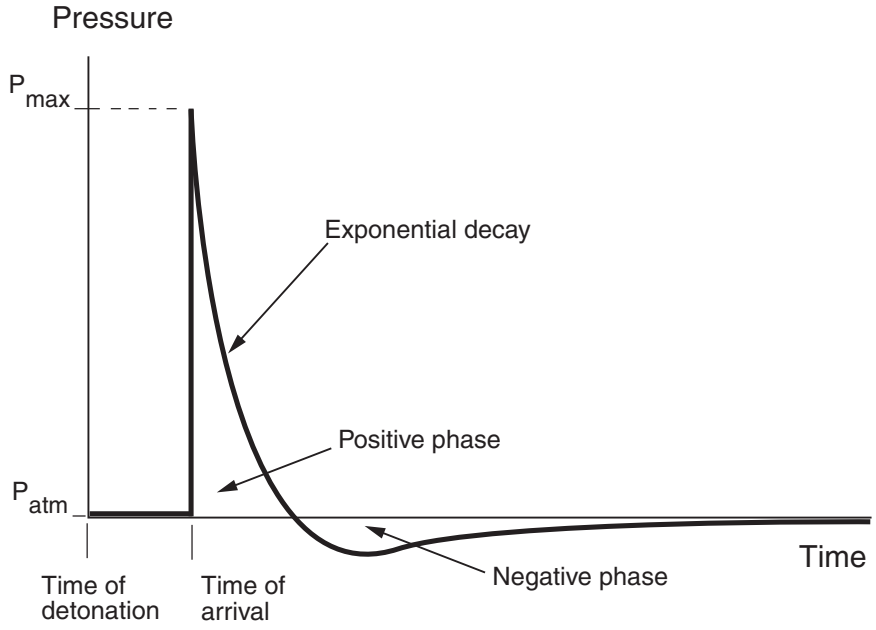


Figure 32.4.6–3 Pressure history of a blast wave.

The CONWEP model uses a scaled distance based on the distance of the loading surface from the source of the explosion and the amount of explosive detonated. For a given scaled distance, the model provides the following empirical data: the maximum overpressure (above atmospheric), the arrival time, the positive phase duration, and the exponential decay coefficient for both the incident pressure and the reflected pressure. Using these parameters, the entire time history of both the incident pressure and reflected pressure as shown in Figure 32.4.6–3 can be constructed. Use of a standoff point is not required.

The total pressure, $P(t)$, on a surface due to the blast wave is a function of the incident pressure, $P_{incident}(t)$, the reflected pressure, $P_{reflect}(t)$, and the angle of incidence, θ, which is defined as the angle between the normal of the loading surface and the vector that points from the surface to the explosion source. The total pressure is defined as

$$P(t) = P_{incident}(t)[1 + \cos\theta - 2\cos^2\theta] + P_{reflect}(t)\cos^2\theta \quad for\ \cos\theta \geq 0;$$

$$P(t) = P_{incident}(t) \quad for\ \cos\theta < 0.$$

The air blast loading due to the total pressure can be scaled using a magnitude scale factor.

A detonation time can be specified if the explosion does not occur at the start of the analysis. The detonation time needs to be given in total time; see "Conventions," Section 1.2.2, for a description of the time convention. The arrival time at a location is defined as the elapsed time for the wave to arrive at that location after detonation.

The CONWEP empirical data are given in a specific set of units, which must be converted to the units used in the analysis. You will need to specify multiplying factors for conversion of these units to SI units. For the specification of the mass of the explosive in TNT equivalence, you can choose any convenient mass unit, which can be different from the mass unit used in the analysis. For computation of the pressure loading, you will need to specify multiplying factors for conversion of length, time, and pressure units used in the analysis to SI units. Some typical conversion multiplier values are given in Table 32.4.6–3.

Table 32.4.6–3 Multipliers used in conjunction with the CONWEP model for conversion to SI units.

Quantity	Unit	SI Unit	Multiplier for conversion to SI
Mass	ton	kg	1000
Mass	lb	kg	0.45359
Length	mm	m	0.001
Length	ft	m	0.3048
Time	msec	sec	0.001
Pressure	MPa	Pa	10^{-6}
Pressure	psi	Pa	6894.8
Pressure	psf	Pa	47.88

For any given amount of explosive, the CONWEP empirical data are valid only within a range of distances from the source. The minimum distance at which the data are valid corresponds to the charge radius. Thus, the analysis terminates if the distance of any part of the loading surface from the source is less than the charge radius. For distances that are larger than the maximum valid range, linear extrapolation is used up to an extended maximum range where the reflected pressure decreases to zero. No loading is applied beyond the extended maximum range.

The CONWEP empirical data do not account for shadowing by intervening objects or for any effects due to confinement. In the definition of incident wave interaction using the CONWEP model, you cannot use incident wave reflection.

Input File Usage: Use the following options to specify loading effects due to explosion in air using the CONWEP charge property definition:

∗INCIDENT WAVE INTERACTION PROPERTY,
NAME=*wave property name*, TYPE=AIR BLAST or SURFACE BLAST
∗CONWEP CHARGE PROPERTY
data defining the CONWEP charge
∗INCIDENT WAVE INTERACTION, PROPERTY=*wave property name*, CONWEP
loading surface name, *source node*, *detonation time*, *magnitude scale factor*

Abaqus/CAE Usage: Use the following options to specify loading effects due to explosion in air using the CONWEP charge property definition:

Interaction module: **Create Interaction Property**: **Name**: *wave property name* and **Incident wave**: **Definition**: **Air blast** or **Surface blast**: enter data defining the CONWEP charge

Interaction module: **Create Interaction**: **Name**: *incident wave name* and **Incident wave**: select the source point: **CONWEP (Air/Surface blast)**: select the region: **CONWEP Data**: enter data defining the time of detonation and magnitude scale factor

Modifying or removing incident wave loads

Only the incident wave loads that are specified in a particular step are applied in that step; previous definitions are removed automatically. Consequently, incident wave loads that are active during two subsequent steps should be specified in each step. This is akin to the behavior that can be specified for other types of loads by releasing any load of that type in a step (see "Applying loads: overview," Section 32.4.1).

Alternative incident wave loading interface

In general, the concepts of the alternative incident wave loading interface are the same as the preferred interface; however, the syntax for specifying the incident wave loading is different. The preferred incident wave loading interface is supported in Abaqus/CAE. The alternative interface is not supported in Abaqus/CAE. For conceptual information, see "Incident wave loading due to external sources."

Prescribing the geometric properties and the speed of the incident wave (alternative interface)

Conceptually, the alternative interface is the same as the preferred interface; however, the usages are different. For conceptual information, see "Prescribing geometric properties and the speed of the incident wave."

Input File Usage: ∗INCIDENT WAVE PROPERTY, NAME=*wave property name*, TYPE=PLANE *or* SPHERE
data lines to specify the location of the acoustic source and the standoff point
∗INCIDENT WAVE FLUID PROPERTY
bulk modulus, *mass density*
∗INCIDENT WAVE, PROPERTY=*wave property name*

Abaqus/CAE Usage: The alternative incident wave loading interface is not supported in Abaqus/CAE.

Defining the time history of the source pulse (alternative interface)

Conceptually, the alternative interface is the same as the preferred interface; however, the usages are different. For conceptual information, see "Defining the amplitude of the source pulse."

Input File Usage: Use the following option to define the time history in terms of fluid pressure values:

∗INCIDENT WAVE, PRESSURE AMPLITUDE=*amplitude data table name*
solid or fluid surface name, *reference magnitude*

Use the following option to define the time history in terms of fluid particle acceleration values:

∗INCIDENT WAVE, ACCELERATION AMPLITUDE=*amplitude data table name*
fluid surface name, *reference magnitude*

Abaqus/CAE Usage: The alternative incident wave loading interface is not supported in Abaqus/CAE.

Defining bubble loading for spherical incident wave loading (alternative interface)

Conceptually, the alternative interface is the same as the preferred interface; however, the usages are different. For conceptual information, see "Defining bubble loading for spherical incident wave loading."

To define the bubble dynamics using a model internal to Abaqus, you can specify a bubble amplitude. Use of the bubble loading amplitude is generally similar to the use of any other amplitude in Abaqus.

Input File Usage: Use the following options:

∗AMPLITUDE, DEFINITION=BUBBLE, NAME=*name*
∗INCIDENT WAVE PROPERTY, TYPE=SPHERE,
NAME=*wave property name*
∗INCIDENT WAVE, PRESSURE AMPLITUDE=*name*
solid or fluid surface name, *reference magnitude*

Abaqus/CAE Usage: The alternative incident wave loading interface is not supported in Abaqus/CAE.

To define the bubble dynamics using tabulated data for the pressure and source migration, you can specify independent amplitude curves for the pressure at the standoff point and any source location time histories. The source location amplitude names, or floating point data for source point coordinates that remain fixed, are referred to in the incident wave property definition. The amplitude name for the pressure amplitude is referred to in the incident wave loading definition in the usual manner.

Input File Usage: Use the following options:

∗AMPLITUDE, DEFINITION=TABULAR, NAME=*Pressure*
∗AMPLITUDE, DEFINITION=TABULAR, NAME=*X*
∗AMPLITUDE, DEFINITION=TABULAR, NAME=*Y*
∗AMPLITUDE, DEFINITION=TABULAR, NAME=*Z*
∗INCIDENT WAVE PROPERTY, TYPE=SPHERE,
NAME=*wave property name*
{standoff point data}
X, *Y*, *Z*
∗INCIDENT WAVE, PRESSURE AMPLITUDE=*Pressure*
solid or fluid surface name, *reference magnitude*

Abaqus/CAE Usage: The alternative incident wave loading interface is not supported in Abaqus/CAE.

Specifying the reflection effects (alternative interface)

Conceptually, the alternative interface is the same as the preferred interface; however, the usages are different. For conceptual information, see “Specifying the reflection effects.”

Input File Usage: Use the following option in conjunction with the ∗INCIDENT WAVE option to define an incident wave reflection plane:

∗INCIDENT WAVE REFLECTION

Abaqus/CAE Usage: The alternative incident wave loading interface is not supported in Abaqus/CAE.

Modeling incident wave loading on a moving structure (alternative interface)

To model the effect of rigid motion of a structure such as a ship during the incident wave loading history, the standoff point can have a specified velocity. It is assumed that the entire fluid-solid model is moving at this velocity with respect to the source point during the loading and that the speed of the model’s motion is low compared to the speed of propagation of the incident wave.

Input File Usage: ∗INCIDENT WAVE PROPERTY, NAME=*wave property name*
data line to specify the velocity of the standoff point

Abaqus/CAE Usage: The alternative incident wave loading interface is not supported in Abaqus/CAE.

Example: submarine close to the free surface

The problem shown in Figure 32.4.6–4 has the following features: a free surface A_0, seabed A_{sb} as a reflection plane, a wet solid surface A_{sw}, the fluid surface A_{fw} that is tied to the solid surface A_{sw}, and the boundary $A_{\inf}$ of the finite modeled domain separating the infinite acoustic medium. The source S of the underwater explosion loading is also shown.

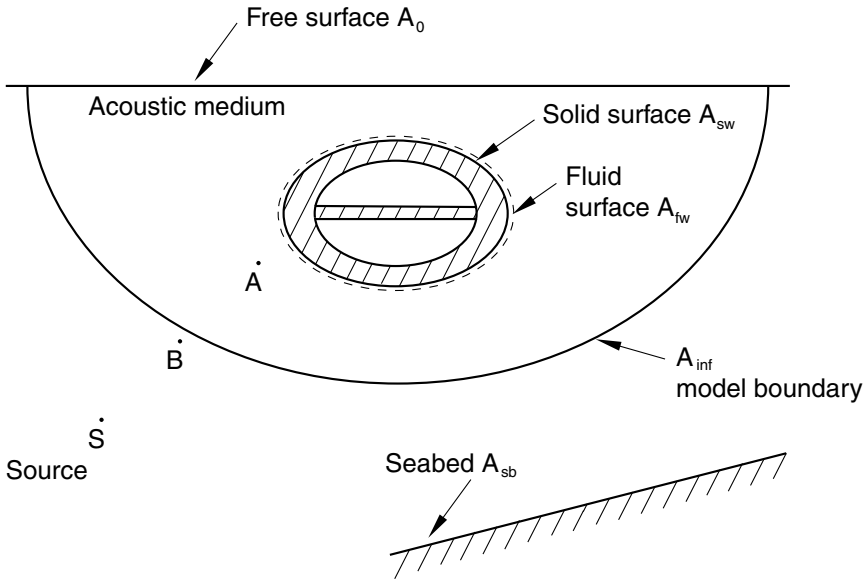


Figure 32.4.6–4 Incident wave loading on a submarine lying near a free surface.

Scattered wave solution

Here the scattered wave response in the acoustic medium is of interest along with that of the structure to the incident wave loading. Cavitation in the fluid is not considered in a scattered wave formulation. Similarly, the initial hydrostatic pressure in the fluid is not modeled.

The zero dynamic acoustic pressure boundary condition on the free surface requires both a "soft" reflection plane coinciding with the free surface A_0 and a zero scattered pressure boundary condition at the nodes on this free surface. The incident wave loading is applied on the fluid surface, A_{fw}, and on the wet solid surface, A_{sw}. The incident wave loading can be only of pressure amplitude type since the loading includes a solid surface.

A good location for the standoff node is marked as A in Figure 32.4.6–4. This node is in the fluid, close to the structure, and closer to incident wave source S than any portion of the seabed or the free surface. The standoff node's offset from the loaded surfaces is exaggerated for emphasis in the figure.

The radiation condition is specified on the acoustic surface A_{inf} such that the scattered wave impinging on this boundary with the infinite medium does not reflect back into the computational domain. The seabed is modeled with an incident wave reflection plane on surface A_{sb}. The reflection loss at this seabed surface is modeled using an impedance property.

If the response of the structure in the nonlinear regime is of interest, the initial stress state in the structure should be established using Abaqus/Standard in a static analysis. The stress state in the structure is then imported into Abaqus/Explicit, and the loading on the solid surfaces causing the initial stress state is respecified in the acoustic analysis.

The following template schematically shows some of the Abaqus input file options that are used to solve this problem using the scattered wave formulation:

***HEADING**
…
***SURFACE, NAME=**A_{fw}
Data lines to define the acoustic surface that is wetting the solid
***SURFACE, NAME=**A_{sw}
Data lines to define the solid surface that is wetted by the fluid
***SURFACE, NAME=**A_{inf}
Data lines to define the acoustic surface separating the modeled region from the infinite medium
***INCIDENT WAVE INTERACTION PROPERTY, NAME=IWPROP**
***AMPLITUDE, DEFINITION=TABULAR, NAME=PRESSUREVTIME**
***TIE, NAME=COUPLING**
A_{fw}, A_{sw}
***STEP**
**** For an Abaqus/Standard analysis:**
***DYNAMIC**
**** For an Abaqus/Explicit analysis:**
***DYNAMIC, EXPLICIT**
**** Load the acoustic surface**
***INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=PRESSUREVTIME, PROPERTY=IWPROP**
A_{fw}*, source node, standoff node, reference magnitude*
***INCIDENT WAVE REFLECTION**
Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
***INCIDENT WAVE REFLECTION**
Data lines for a "soft" reflection plane over the free surface A_0.
**** Load the solid surface**
***INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=PRESSUREVTIME, PROPERTY=IWPROP**
A_{sw}*, source node, standoff node, reference magnitude*
***INCIDENT WAVE REFLECTION**
Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
***INCIDENT WAVE REFLECTION**

Data lines for a "soft" reflection plane over the free surface A_0.
***BOUNDARY**
**** zero pressure boundary condition on the free surface**
Set of nodes on the free surface A_0, 8, 8, 0.0
***SIMPEDANCE**
A_{inf},
***END STEP**

Total wave solution

Here the total wave response in the acoustic medium is of interest along with that of the structure to the incident wave loading. Cavitation in the fluid may be included. Similarly, a linearly varying initial hydrostatic pressure in the fluid can be specified.

The zero dynamic acoustic pressure boundary condition on the free surfaces requires only a zero pressure boundary condition at the nodes on this free surface. A reflection plane should not be included along the free surface. The incident wave loading is applied only on the fluid surface, A_{inf}, that separates the modeled region from the surrounding infinite acoustic medium. No incident wave should be applied directly on the structure surfaces. If the incident wave is considered planar, an acceleration-type amplitude can be used with the incident wave loading. Otherwise, a pressure-type amplitude must be used with the incident wave loading.

An ideal location for the standoff node depends on the type of amplitude used for the time history of the incident wave loading. The location ***A*** shown in Figure 32.4.6–4 can be used if the incident wave loading time history is of pressure amplitude type. Otherwise, the location ***B*** that is just on the boundary A_{inf} and closer to the source ***S*** than any part of either the seabed or the free surface can be used.

The nonreflecting impedance condition is specified on the acoustic surface, A_{inf}, such that the scattered part of the total wave impinging on this boundary with the infinite medium does not reflect back into the computational domain. The seabed is modeled with an incident wave reflection plane on the surface A_{sb}.

If the response of the structure in the nonlinear regime is of interest, the initial stress state in the structure should be established using Abaqus/Standard in a static analysis. The stress state in the structure is then imported into Abaqus/Explicit, and the loading on the solid surfaces causing the initial stress state is respecified in the acoustic analysis.

The following template schematically shows some of the input file options that are used to solve this problem using the total wave formulation:

***HEADING**
…
***ACOUSTIC WAVE FORMULATION, TYPE=TOTAL WAVE**
***MATERIAL, NAME=CAVITATING_FLUID**
***ACOUSTIC MEDIUM, BULK MODULUS**
Data lines to define the fluid bulk modulus
***ACOUSTIC MEDIUM, CAVITATION LIMIT**
Data lines to define the fluid cavitation limit

...
***SURFACE, NAME=**A_{fw}
Data lines to define the acoustic surface that is wetting the solid
***SURFACE, NAME=**A_{sw}
Data lines to define the solid surface that is wetted by the fluid
***SURFACE, NAME=**$A_{\inf}$
Data lines to define the acoustic surface separating the modeled region from the infinite medium
***INCIDENT WAVE INTERACTION PROPERTY, NAME=IWPROP**
***AMPLITUDE, DEFINITION=TABULAR, NAME=PRESSUREVTIME**
Data lines to define the pressure-time history at the standoff point
***TIE, NAME=COUPLING**
A_{fw}, A_{sw}
***INITIAL CONDITIONS, TYPE=ACOUSTIC STATIC PRESSURE**
Data lines to define the initial linear hydrostatic pressure in the fluid
***STEP**
***DYNAMIC, EXPLICIT**
**** Load the acoustic surface**
***INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=PRESSUREVTIME, PROPERTY=IWPROP**
$A_{\inf}$*, source node, standoff node, reference magnitude*
***INCIDENT WAVE REFLECTION**
Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
***BOUNDARY**
**** zero pressure boundary condition on the free surface**
Set of nodes on the free surface A_0*, 8, 8, 0.0*
***SIMPEDANCE**
$A_{\inf}$,
***END STEP**

Example: submarine in deep water

This problem is similar to the previous example of a submarine close to the free surface except for the following differences. There is no free surface in this problem; and the fluid surface, $A_{\inf}$, and the fluid medium completely enclose the structure. If the structure is sufficiently deep in the water, hydrostatic pressure may be considered uniform instead of varying linearly with depth. Under this assumption, the initial stress state in the structure can be established with a uniform pressure loading all around it, if desired. In addition, if the structure is sufficiently deep in the water, the hydrostatic pressure may be significant compared to the incident wave loading; hence, the cavitation in the fluid may not be of concern.

Example: surface ship

Here the effect of underwater explosion loading on a surface ship is of interest (see Figure 32.4.6–5). This problem is similar to the previous example of a submarine close to the free surface except for the

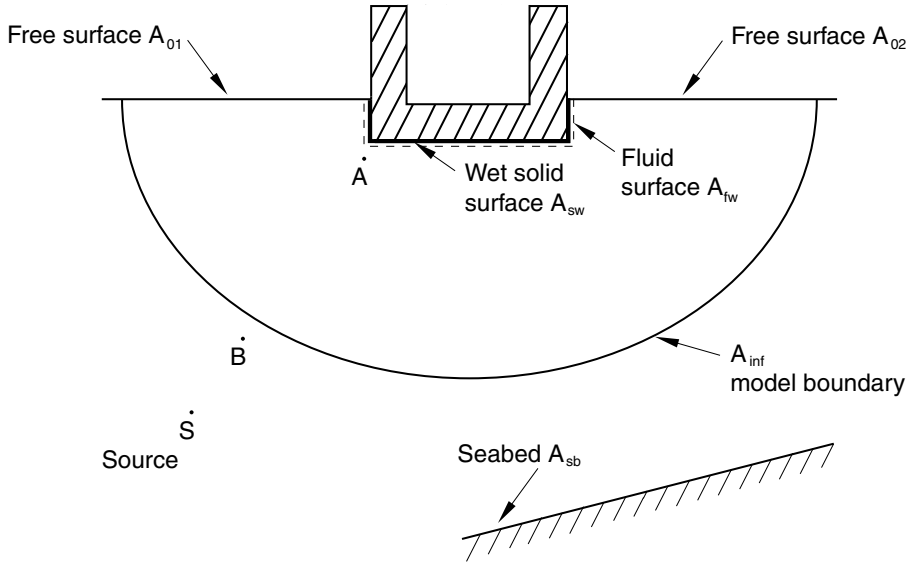


Figure 32.4.6–5 Modeling of incident wave loading on a surface ship.

following differences. The free surface of fluid is not continuous, and a part of the structure is exposed to the atmosphere. A soft reflection plane coinciding with the free surface is not used in this problem as in the submarine problems under the scattered wave formulation. To be able to use the scattered wave formulation in this case, the modeling technique is used in which the free surface is replaced with "structural fluid" elements. A layer of fluid at the free surface is modeled using non-acoustic elements such as membrane elements. These elements are coupled to the underlying acoustic fluid using a mesh tie constraint. The non-acoustic elements have properties similar to the fluid itself since these elements are replacing the fluid medium near the free surface and should have a thickness similar to the height of the adjacent acoustic elements. Incident wave loading with the scattered wave formulation must now be applied on these newly created surfaces as well. This technique has the added advantage of providing the deformed shape of the free surface under the loading.

The following template shows some of the Abaqus input file options used for this case:

```
*HEADING
…
*SURFACE, NAME=A01_structuralfluid
Data lines to define the "structural fluid" surface
*SURFACE, NAME=A01_acousticfluid
Data lines to define the adjacent acoustic fluid surface
*SURFACE, NAME=A02_structuralfluid
Data lines to define the "structural fluid" surface
*SURFACE, NAME=A02_acousticfluid
```

Data lines to define the adjacent acoustic fluid surface
***SURFACE, NAME=Asw_solid**
Data lines to define the actual solid surface that is wetted by the fluid
***SURFACE, NAME=Asw_fluid**
Data lines to define the actual acoustic surface that is adjacent to the structure
***SURFACE, NAME=**$A_{\inf}$
Data lines to define the acoustic surface separating the modeled region from the infinite medium
***INCIDENT WAVE INTERACTION PROPERTY, NAME=IWPROP**
***AMPLITUDE, DEFINITION=TABULAR, NAME=PRESSUREVTIME**
Data lines to define the pressure-time history at the standoff point
***TIE, NAME=COUPLING**
Asw_fluid, Asw_solid
A01_acousticfluid, A01_structuralfluid
A02_acousticfluid, A02_structuralfluid
***STEP**
**** For an Abaqus/Standard analysis:**
***DYNAMIC**
**** For an Abaqus/Explicit analysis:**
***DYNAMIC, EXPLICIT**
**** Load the acoustic surfaces**
***INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=PRESSUREVTIME, PROPERTY=IWPROP**
A01_acousticfluid, source point, standoff point, reference magnitude
***INCIDENT WAVE REFLECTION**
Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
***INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=PRESSUREVTIME, PROPERTY=IWPROP**
A02_acousticfluid, source point, standoff point, reference magnitude
***INCIDENT WAVE REFLECTION**
Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
***INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=PRESSUREVTIME, PROPERTY=IWPROP**
Asw_fluid, source point, standoff point, reference magnitude
***INCIDENT WAVE REFLECTION**
Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
**** Load the solid surfaces**
***INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=PRESSUREVTIME, PROPERTY=IWPROP**
A01_structuralfluid, source point, standoff point, reference magnitude
***INCIDENT WAVE REFLECTION**
Data lines for the reflection plane over the seabed A_{sb}*, seabed_Q*
***INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=PRESSUREVTIME,**

PROPERTY=IWPROP
A02_structuralfluid, source point, standoff point, reference magnitude
***INCIDENT WAVE REFLECTION**
Data lines for the reflection plane over the seabed A_{sb}, seabed_Q
***INCIDENT WAVE INTERACTION, PRESSURE AMPLITUDE=PRESSUREVTIME, PROPERTY=IWPROP**
Asw_solid, source point, standoff point, reference magnitude
***INCIDENT WAVE REFLECTION**
Data lines for the reflection plane over the seabed A_{sb}, seabed_Q
***SIMPEDANCE**
A_{inf},
***END STEP**

Compared to the total wave formulation analysis of a submarine close to the free surface, the following differences are noteworthy. As shown in Figure 32.4.6–5, the free surface with zero dynamic pressure boundary condition is now split into two parts: A_{01} and A_{02}. The fluid surface wetting the ship (A_{fw}) and the wetted ship surface (A_{sw}), which are tied together, do not encircle the whole structure. Besides these differences, the modeling considerations for the surface ship problem are similar to the total wave analysis of the submarine near the free surface.

Example: airblast loading on a structure

Here the effect of airblast (explosion in the air) loading on a structure is of interest (see Figure 32.4.6–6). Since the stiffness and inertia of the air medium are negligible, the acoustic medium is not modeled. Rather the incident wave loading is applied directly on the structure itself. The solid surface A_{sw} where the incident wave loading is applied is shown in Figure 32.4.6–6. Since the acoustic medium is not modeled, the total wave and the scattered wave formulations are identical.

Example: fluid cavitation without incident wave loading

You may be interested in modeling acoustic problems in Abaqus/Explicit where the loading is applied through either prescribed pressure boundaries or specified pressure-conjugate concentrated loads. Choice of the scattered or the total wave formulation is not relevant in these problems even when the acoustic medium is capable of cavitation.

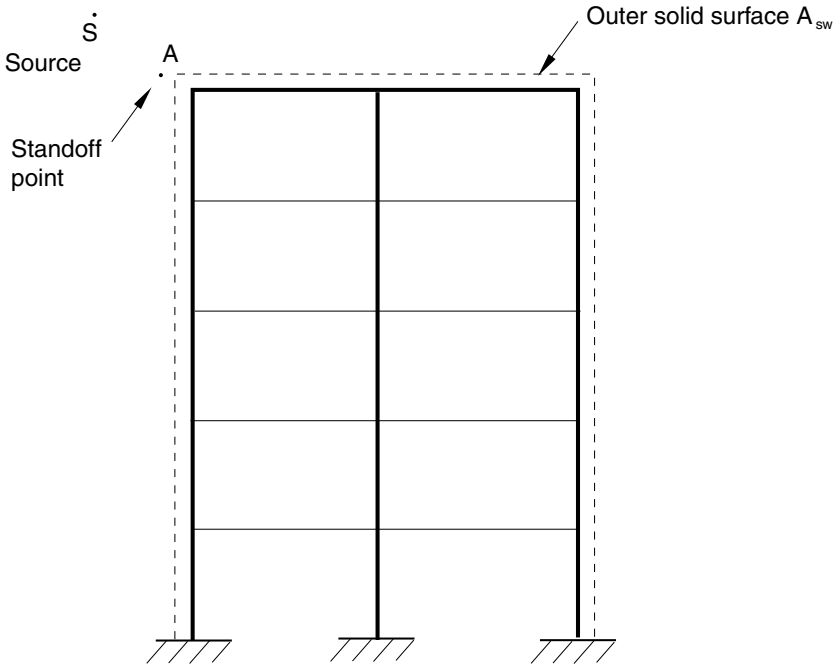


Figure 32.4.6–6 Modeling of airblast loading on a structure.

32.4.7 PORE FLUID FLOW

Products: Abaqus/Standard Abaqus/CAE

References

- "Applying loads: overview," Section 32.4.1
- *CFLOW
- *DFLOW
- *DSFLOW
- *FLOW
- *SFLOW
- "Defining a surface pore fluid flow," Section 16.9.22 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a concentrated pore fluid flow," Section 16.9.21 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Pore fluid flow can be prescribed in coupled pore fluid diffusion/stress analysis (see "Coupled pore fluid diffusion and stress analysis," Section 6.8.1) and in the geostatic stress field procedure (see "Geostatic stress state," Section 6.8.2). Pore fluid flow can be prescribed by:

- defining seepage coefficients and sink pore pressures on element faces or surfaces;
- defining drainage-only seepage coefficients on element faces or surfaces that are applied only when surface pore pressures are positive; or
- prescribing an outward normal flow velocity directly at nodes, on element faces, or on surfaces.

Defining pore fluid flow as a function of the current pore pressure in consolidation analysis

In consolidation analysis you can provide seepage coefficients and sink pore pressures on element faces or surfaces to control normal pore fluid flow from the interior of the region modeled to the exterior of the region.

The surface condition assumes that the pore fluid flows in proportion to the difference between the current pore pressure on the surface, u_w, and some reference value of pore pressure, u_w^∞:

$$v_n = k_s(u_w - u_w^\infty),$$

where

v_n is the component of the pore fluid velocity in the direction of the outward normal to the surface;

k_s is the seepage coefficient;

u_w is the current pore pressure at this point on the surface; and

u_w^∞ is a reference pore pressure value.

Specifying element-based pore fluid flow

To define element-based pore fluid flow, specify the element or element set name; the distributed load type; the reference pore pressure, u_w^∞; and the reference seepage coefficient, k_s. The face of the elements upon which the normal flow is enforced is identified by a seepage distributed load type. The seepage types available depend on the element type (see Part VI, "Elements").

Input File Usage: ∗FLOW
element number or element set name, Q*n*, u_w^∞, k_s

Abaqus/CAE Usage: Pore fluid flow cannot be defined as a function of the current pore pressure in Abaqus/CAE.

Specifying surface-based pore fluid flow

To define surface-based pore fluid flow, specify a surface name, the seepage flow type, the reference pore pressure, and the reference seepage coefficient. The element-based surface (see "Element-based surface definition," Section 2.3.2) contains the element and face information.

Input File Usage: ∗SFLOW
surface name, Q, u_w^∞, k_s

Abaqus/CAE Usage: Pore fluid flow cannot be defined as a function of the current pore pressure in Abaqus/CAE.

Defining drainage-only flow

Drainage-only flow types can be specified for element-based or surface-based pore fluid flow to indicate that normal pore fluid flow occurs only from the interior to the exterior region of the model. The drainage-only flow surface condition assumes that the pore fluid flows in proportion to the magnitude of the current pore pressure on the surface, u_w, when that pressure is positive:

$$v_n = k_s u_w, \quad u_w > 0$$
$$v_n = 0, \qquad u_w \le 0,$$

where

v_n is the component of the pore fluid velocity in the direction of the outward normal to the surface;

k_s is the seepage coefficient; and

u_w is the current pore pressure at this point on the surface.

Figure 32.4.7–1 illustrates this pore pressure–velocity relationship. This surface condition is designed for use with the total pore pressure formulation (see "Coupled pore fluid diffusion and stress analysis," Section 6.8.1), mainly for cases where the phreatic surface intersects an exterior surface that is free to drain. See "Calculation of phreatic surface in an earth dam," Section 10.1.2 of the Abaqus Example Problems Manual, for an example of this type of calculation.

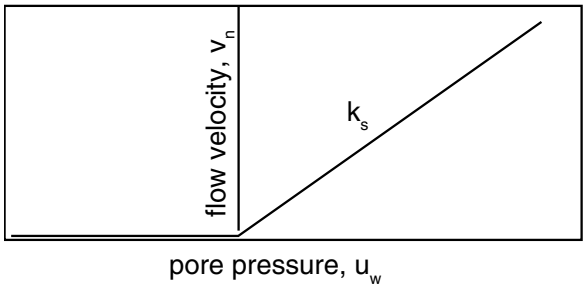


Figure 32.4.7–1 Drainage-only pore pressure–velocity relationship.

When surface pore pressures are negative, the constraint will properly enforce the condition that no fluid can enter the interior region. When surface pore pressures are positive, the constraint will permit fluid flow from the interior to the exterior region of the model. When the seepage coefficient value, k_s, is large, this flow will approximately enforce the requirement that the pore pressure should be zero on a freely draining surface. To achieve this condition, it is necessary to choose the value of k_s to be much larger than a characteristic seepage coefficient for the material in the underlying elements:

$$k_s \gg k/\gamma_w c,$$

where

- k is the permeability of the underlying material;
- γ_w is the fluid specific weight; and
- c is a characteristic length of the underlying elements.

Values of $k_s \approx 10^5 k/\gamma_w c$ will be adequate for most analyses. Larger values of k_s could result in poor conditioning of the model. In all cases the freely draining flow type represents discontinuously nonlinear behavior, and its use may require appropriate solution controls (see "Commonly used control parameters," Section 7.2.2).

Input File Usage: Use the following option to define element-based drainage-only flow:

```
*FLOW
element number or element set name, QnD, u_w^∞, k_s
```

Use the following option to define surface-based drainage-only flow:

```
*SFLOW
surface name, QD, u_w^∞, k_s
```

Abaqus/CAE Usage: Pore fluid flow cannot be defined as a function of the current pore pressure in Abaqus/CAE.

Modifying or removing seepage coefficients and reference pore pressures

Seepage coefficients and reference pore pressures can be added, modified, or removed as described in "Applying loads: overview," Section 32.4.1.

Specifying a time-dependent reference pore pressure

The magnitude of the reference pore pressure, u_w^∞, can be controlled by referring to an amplitude curve. If different variations are needed for different portions of the flow, repeat the flow definition with each referring to its own amplitude curve. See "Applying loads: overview," Section 32.4.1, and "Amplitude curves," Section 32.1.2, for details.

Defining nonuniform flow in a user subroutine

To define nonuniform flow, the variation of the reference pore pressure and the seepage coefficient as functions of position, time, pore pressure, etc. can be defined in user subroutine **FLOW**.

Input File Usage: Use the following option to define a nonuniform element-based flow:

*FLOW
element number or element set name, Q*n*NU

Use the following option to define a nonuniform surface-based flow:

*SFLOW
surface name, QNU

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

Prescribing seepage flow velocity and seepage flow directly in consolidation analysis

You can directly prescribe an outward normal flow velocity, v_n, across a surface or an outward normal flow at a node in consolidation analysis.

Prescribing element-based seepage flow velocity

To prescribe an element-based seepage flow velocity, specify the element or element set name, the seepage type, and the outward normal flow velocity. The face of the element for which the seepage flow is being defined is identified by the seepage type. The seepage types available depend on the element type (see Part VI, "Elements").

Input File Usage: *DFLOW
element number or element set name, S*n*, v_n

Abaqus/CAE Usage: Load module: **Create Load**: choose **Fluid** for the **Category** and **Surface pore fluid** for the **Types for Selected Step**: select region: **Distribution**: select an analytical field, **Magnitude**: v_n

Prescribing surface-based seepage flow velocity

To prescribe a surface-based seepage flow velocity, specify a surface name, the seepage flow type, and the pore fluid velocity. The element-based surface (see "Element-based surface definition," Section 2.3.2) contains the element and face information.

Input File Usage: *DSFLOW
surface name, S, v_n

Abaqus/CAE Usage: Load module: **Create Load**: choose **Fluid** for the **Category** and **Surface pore fluid** for the **Types for Selected Step**: select region: **Distribution**: **Uniform**, **Magnitude**: v_n

Prescribing node-based seepage flow

To prescribe node-based seepage flow, specify the node or node set name and the magnitude of the flow per unit time.

Input File Usage: *CFLOW
node number or node set name, , *magnitude of the flow per unit time*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Fluid** for the **Category** and **Concentrated pore fluid** for the **Types for Selected Step**: select region: **Magnitude**: *magnitude of the flow per unit time*

Modifying or removing seepage flow velocities and seepage flow

Seepage flow velocities can be added, modified, or removed as described in "Applying loads: overview," Section 32.4.1.

Specifying time-dependent flow velocity and flow

The magnitude of the seepage velocity, v_n, can be controlled by referring to an amplitude curve. To specify different variations for different flows, repeat the seepage flow velocity or seepage flow definition with each referring to its own amplitude curve. See "Applying loads: overview," Section 32.4.1, and "Amplitude curves," Section 32.1.2, for details.

Defining nonuniform flow velocities in a user subroutine

To define nonuniform element-based or surface-based flow, the variation of the seepage magnitude as a function of position, time, pore pressure, etc. can be defined in user subroutine **DFLOW**. If the optional seepage velocity, v_n, is specified directly, this value is passed into user subroutine **DFLOW** in the variable used to define the seepage magnitude.

Input File Usage: Use the following option to define nonuniform element-based flow:

*DFLOW
element number or element set name, S*n*NU, v_n

Use the following option to define nonuniform surface-based flow:

*DSFLOW
surface name, SNU, v_n

Abaqus/CAE Usage: Use the following input to define nonuniform surface-based flow:

Load module: **Create Load**: choose **Fluid** for the **Category** and **Surface pore fluid** for the **Types for Selected Step**: select region: **Distribution: User-defined**, **Magnitude**: v_n

Nonuniform element-based flow is not supported in Abaqus/CAE.

32.5 Prescribed assembly loads

- “Prescribed assembly loads,” Section 32.5.1

32.5.1 PRESCRIBED ASSEMBLY LOADS

Products: Abaqus/Standard Abaqus/CAE

References

- "Prescribed conditions: overview," Section 32.1.1
- *BOUNDARY
- *CLOAD
- *PRE-TENSION SECTION
- *SURFACE
- Chapter 22, "Bolt loads," of the Abaqus/CAE User's Manual

Overview

Assembly loads:

- can be used to simulate the loading of fasteners in a structure;
- are applied across user-defined pre-tension sections;
- are applied to pre-tension nodes that are associated with the pre-tension sections; and
- require the specification of pre-tension loads or tightening adjustments.

Concept of an assembly load

Figure 32.5.1–1 is a simple example that illustrates the concept of an assembly load.

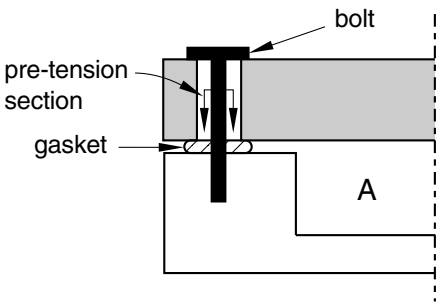


Figure 32.5.1–1 Example of assembly load.

Container A is sealed by pre-tensioning the bolts that hold the lid, which places the gasket under pressure. This pre-tensioning is simulated in Abaqus/Standard by adding a "cutting surface," or pre-tension section, in the bolt, as shown in Figure 32.5.1–1, and subjecting it to a tensile load. By modifying the elements on

one side of the surface, Abaqus/Standard can automatically adjust the length of the bolt at the pre-tension section to achieve the prescribed amount of pre-tension. In later steps further length changes can be prevented so that the bolt acts as a standard, deformable component responding to other loadings on the assembly.

Modeling an assembly load

Abaqus/Standard allows you to prescribe assembly loads across fasteners that are modeled by continuum, truss, or beam elements. The steps needed to model an assembly load vary slightly depending on the type of elements used to model the fasteners.

Modeling a fastener with continuum elements

In continuum elements the pre-tension section is defined as a surface inside the fastener that "cuts" it into two parts (see Figure 32.5.1–2). The pre-tension section can be a group of surfaces for cases where a fastener is composed of several segments.

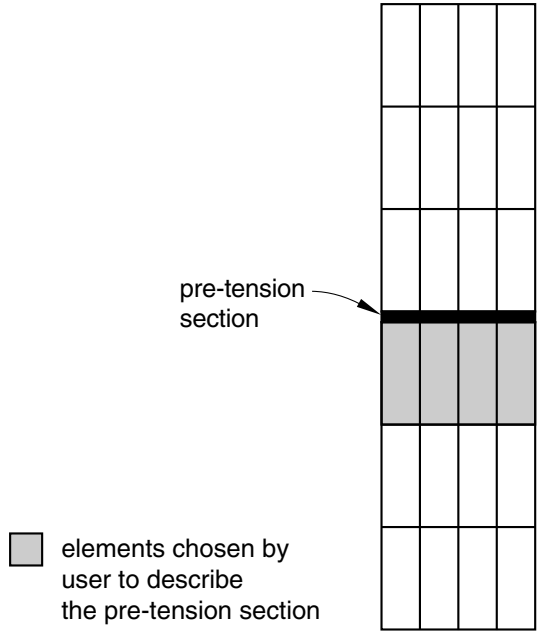


Figure 32.5.1–2 Pre-tension section defined using continuum elements.

The element-based surface contains the element and face information (see "Element-based surface definition," Section 2.3.2). You must convert the surface into a pre-tension section across which pre-tension loads can be applied and assign a controlling node to the pre-tension section.

Input File Usage: Use the following options to model an assembly load across a fastener that is modeled with continuum elements:

*SURFACE, TYPE=ELEMENT, NAME=*surface_name*
*PRE-TENSION SECTION, SURFACE=*surface_name*, NODE=*n*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Bolt load** for the **Types for Selected Step**

Assigning a controlling node to the pre-tension section

The assembly load is transmitted across the pre-tension section by means of the pre-tension node. The pre-tension node should not be attached to any element in the model. It has only one degree of freedom (degree of freedom 1), which represents the relative displacement at the two sides of the cut in the direction of the normal (see Figure 32.5.1–3). The coordinates of this node are not important.

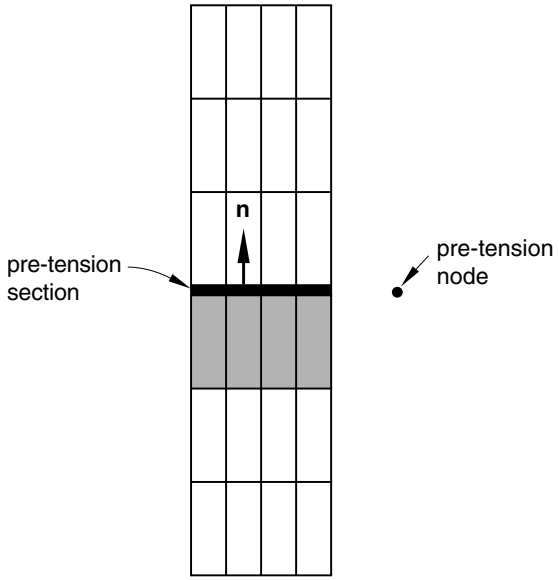


Figure 32.5.1–3 Normal to the pre-tension section; this normal should face away from the underlying elements.

Defining the normal to the pre-tension section

Abaqus/Standard computes an average normal to the section—in the positive surface direction, facing away from the continuum elements used to generate the surface—to determine the direction along which the pre-tension is applied. You may also specify the normal directly (when the desired direction of loading is different from the average normal to the pre-tension section). The normal is not updated when performing large-displacement analysis.

Recognizing elements on either side of the pre-tension section

For all the elements that are connected to the pre-tension section by at least one node, Abaqus/Standard must determine on which side of the pre-tension section each element is located. This process is crucial for the prescribed assembly load to work properly.

The elements used to define the section are referred to as "base elements" in this discussion. All elements on the same side of the section as the base elements are referred to as the "underlying elements." All elements connected to the section that share faces (or in two-dimensional problems, edges) with the base elements are added to the list of underlying elements. This is a repetitive process that enables Abaqus/Standard to find the underlying elements in almost all meshes—triangles; wedges; tetrahedra; and embedded beams, trusses, shells, and membranes—that were not used in the definition of the surface (see Figure 32.5.1–4).

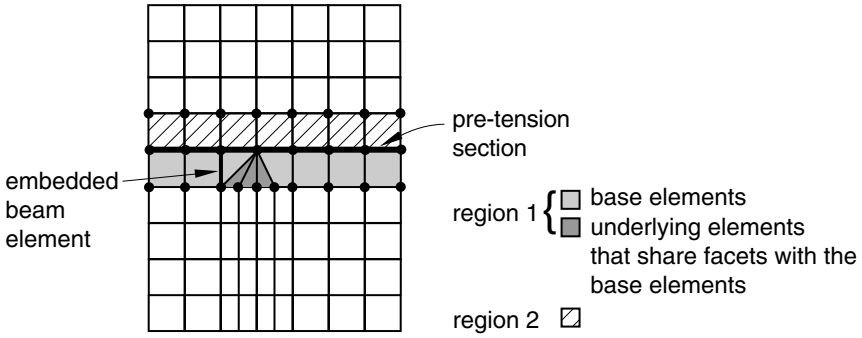


Figure 32.5.1–4 The base elements are used to find the underlying elements.

In most cases this process will group all of the elements that are connected to the section into two regions, as shown in the figure. In rare instances this process may group the elements in more than two regions, in particular if line elements cross over element boundaries. An example is shown in Figure 32.5.1–5; it has three regions, where region 1 is the underlying region. For each region other than region 1 an additional step is necessary to determine on which side of the section the region is located. Abaqus/Standard computes an average normal, $\mathbf{n}$, for all the nodes of the region that belong to the section; it also computes an average position ($\mathbf{A}$) of all these nodes. In addition, it computes an average position ($\mathbf{B}$) of the remaining nodes of the region. If the dot product between the normal $\mathbf{n}$ and the vector AB is negative, the region is assumed to be an underlying region and is added to region 1. This additional step is illustrated in Figure 32.5.1–5 for regions 2 and 3.

This additional step produces an incorrect separation for the beam element shown in Figure 32.5.1–6 since the beam is not found to be an underlying element. If the pre-tension section has an odd shape and one or more line elements that cross over element boundaries are connected to it, consult the list of the underlying elements given in the data (`.dat`) file to make sure that the underlying elements are listed correctly.

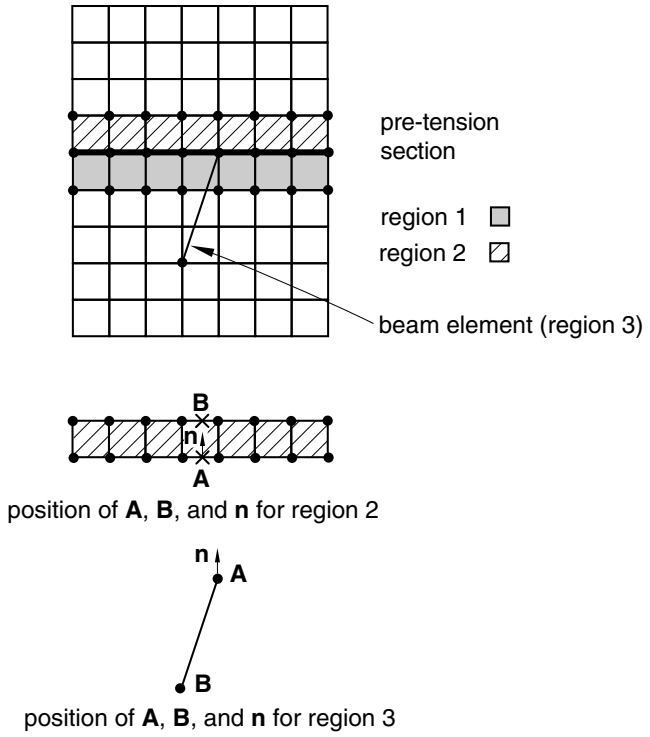


Figure 32.5.1–5 An additional underlying element is found.

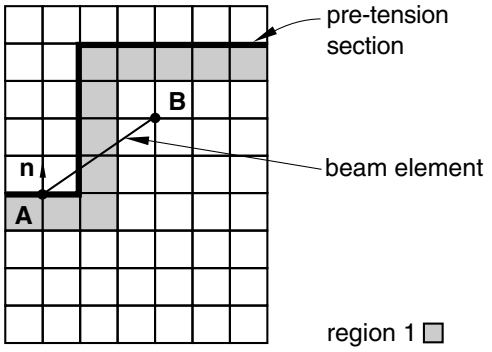


Figure 32.5.1–6 An additional underlying element is not found.

Elements that are connected only to the nodes on the pre-tension section, including single-node elements (such as SPRING1, DASHPOT1, and MASS elements) are not included as underlying elements: they are considered to be attached to the other side of the section.

Modeling a fastener with truss or beam elements

When a pre-tensioned component is modeled with truss or beam elements, the pre-tension section is reduced to a point. The section is assumed to be located at the last node of the element as defined by the element connectivity (see "Beam element library," Section 28.3.8, and "Truss element library," Section 28.2.2, for a definition of the node ordering for beam and truss elements, respectively), with its normal along the element directed from the first to the last node. As a result, the section is defined entirely by just specifying the element to which an assembly load must be prescribed and associating it with a pre-tension node.

Input File Usage: Use the following option to model an assembly load across fasteners modeled with beam or truss elements:

∗PRE-TENSION SECTION, ELEMENT=*element_number*, NODE=*n*

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Bolt load** for the **Types for Selected Step**

As in the case of a surface-based pre-tension section, the node has only one degree of freedom (degree of freedom 1), which represents the relative displacement on the two sides of the cut in the direction of the normal (see Figure 32.5.1–7). The coordinates of the node are not important.

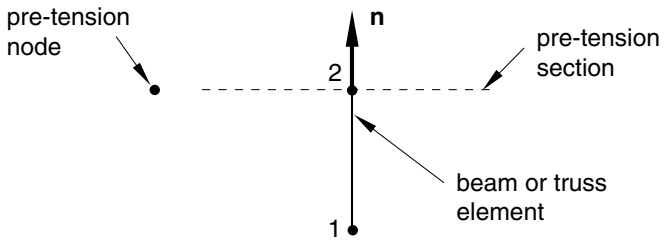


Figure 32.5.1–7 Pre-tension section defined using a truss or beam element.

Defining the normal to the pre-tension section

Abaqus/Standard computes the normal as the vector from the first to the last node in the connectivity of the underlying element. Alternatively, you can specify the normal to the section directly. This normal is not updated during large-displacement analysis.

Defining multiple pre-tension sections

You can define multiple pre-tension sections by repeating the pre-tension section definition input. Each pre-tension section should have its own pre-tension node.

Use with nodal transformations

A local coordinate system (see "Transformed coordinate systems," Section 2.1.5) cannot be used at a pre-tension node. It can be used at nodes located on pre-tension sections.

Applying the prescribed assembly load

The pre-tension load is transmitted across the pre-tension section by means of the pre-tension node.

Prescribing the pre-tension force

You can apply a concentrated load to the pre-tension node. This load is the self-equilibrating force carried across the pre-tension section, acting in the direction of the normal on the part of the fastener underlying the pre-tension section (the part that contains the elements that were used in the definition of the pre-tension section; see Figure 32.5.1–8).

Input File Usage: *CLOAD

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Bolt load** for the **Types for Selected Step**: select surface and if, necessary, datum axis: **Method: Apply force**

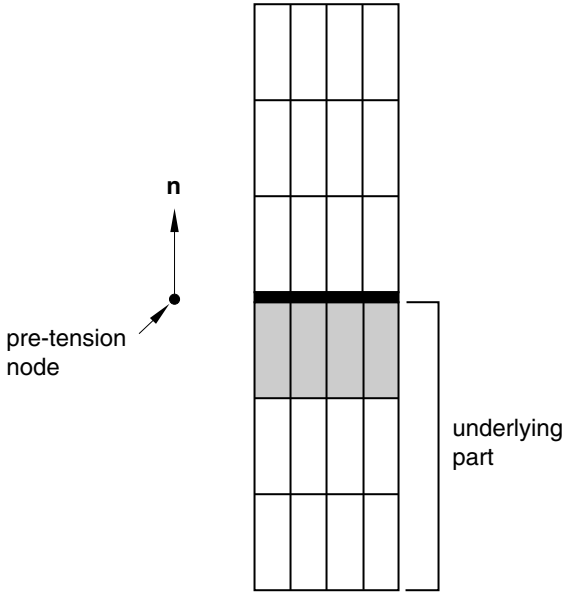


Figure 32.5.1–8 The prescribed assembly load is given at the pre-tension node and applied in direction **n**.

Prescribing a tightening adjustment

You can prescribe a tightening adjustment of the pre-tension section by using a nonzero boundary condition at the pre-tension node (which corresponds to a prescribed change in the length of the component cut by the pre-tension section in the direction of the normal).

Input File Usage: *BOUNDARY

Abaqus/CAE Usage: Load module: **Create Load**: choose **Mechanical** for the **Category** and **Bolt load** for the **Types for Selected Step**: select surface and if, necessary, datum axis: **Method: Adjust length**

Controlling the pre-tension node during the analysis

You can maintain the initial adjustment of the pre-tension section by using a boundary condition fixing the degrees of freedom at their current values at the start of the step once an initial pre-tension is applied in the fastener; this technique enables the load across the pre-tension section to change according to the externally applied loads to maintain equilibrium. If the initial adjustment of a section is not maintained, the force in the fastener will remain constant.

When a pre-tension node is not controlled by a boundary condition, make sure that the components of the structure are kinematically constrained; otherwise, the structure could fall apart due to the presence of rigid body modes. Abaqus/Standard will issue a warning message if it does not find any boundary condition or load on a pre-tension node during the first step of the analysis.

Display of results

Abaqus/Standard automatically adjusts the length of the component at the pre-tension section to achieve the prescribed amount of pre-tension. This adjustment is done by moving the nodes of the underlying elements that lie on the pre-tension section relative to the same nodes when they appear in the other elements connected to the pre-tension section. As a result, the underlying elements will appear shrunk, even though they carry tensile stresses when a pre-tension is applied.

Limitations when using assembly loads

Assembly loads are subject to the following limitations:

- An assembly load cannot be specified within a substructure.
- If a submodeling analysis is performed (“Submodeling: overview,” Section 10.2.1), any pre-tension section should not cross regions where driven nodes are specified. In other words, a pre-tension section should appear either entirely in the region of the global model that is not part of a submodel or entirely in the region of the global model that is part of a submodel. In the latter case, a pre-tension section must also appear in the submodel when the submodel analysis is performed.
- Nodes of a pre-tension section should not be connected to other parts of the body through multi-point constraints (“General multi-point constraints,” Section 33.2.2). These nodes can be connected to other parts of the body through equations (“Linear constraint equations,” Section 33.2.1). However,

an equation connecting a node on the pre-tension section to a node located on the underlying side of the section introduces a constraint that spans across the pre-tension cut and, therefore, interacts directly with the application of the pre-tension load. On the other hand, an equation connecting a node on the pre-tension section to a node on the other side of the section does not influence the application of the pre-tension load.

Procedures

Any of the Abaqus/Standard procedures that use element types with displacement degrees of freedom can be used. Static analysis is the most likely procedure type to be used when prescribing the initial pre-tension ("Static stress analysis," Section 6.2.2). Other analysis types such as coupled temperature-displacement ("Sequentially coupled thermal-stress analysis," Section 6.5.3) or coupled thermal-electrical-structural ("Fully coupled thermal-electrical-structural analysis," Section 6.7.4) can also be used. Once the initial pre-tension is applied, a static or dynamic analysis ("Dynamic analysis procedures: overview," Section 6.3.1) may, for instance, be used to apply additional loads while maintaining the tightening adjustment.

Output

The total force across the pre-tension section is the sum of the reaction force at the pre-tension node plus any concentrated load specified at that node. The total force across the pre-tension section is available as output using the output variable identifier TF (see "Abaqus/Standard output variable identifiers," Section 4.2.1). The forces are along the normal direction. The shear force across the pre-tension section is not available for output.

The tightening adjustment of the pre-tension section is available as the displacement of the pre-tension node. The output of displacement is requested using output identifier U. Only the adjustment normal to the pre-tension section is output since there is no adjustment in any other direction.

The stress distribution across the pre-tension section is not available directly; however, the stresses in the underlying elements can be displayed readily. Alternatively, a tied contact pair can be inserted at the location of the pre-tension section to enable stress distribution output by means of output identifiers CPRESS and CSHEAR. See "Defining tied contact in Abaqus/Standard," Section 34.3.7, for details on defining tied contact.

Input file template

```
*HEADING
Prescribed assembly load; example using continuum elements
...
*NODE
Optionally define the pre-tension node
*SURFACE, NAME=name
Data lines that specify the elements and their associated faces to define the pre-tension section
*PRE-TENSION SECTION, SURFACE=name, NODE=pre-tension_node
**
```

```
*STEP
** Application of the pre-tension across the section
*STATIC
Data line to control time incrementation
*CLOAD
pre-tension_node, 1, pre-tension_value
or
*BOUNDARY,AMPLITUDE=amplitude
pre-tension_node, 1, 1, tightening adjustment
*END STEP
*STEP
** maintain the tightening adjustment and apply new loads
*STATIC or *DYNAMIC
Data line to control time incrementation
*BOUNDARY,FIXED
pre-tension_node, 1, 1
*BOUNDARY
Data lines to prescribe other boundary conditions
*CLOAD or *DLOAD
Data lines to prescribe other loading conditions
...
*END STEP
```

32.6 Predefined fields

- “Predefined fields,” Section 32.6.1

32.6.1 PREDEFINED FIELDS

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Prescribed conditions: overview," Section 32.1.1
- *TEMPERATURE
- *FIELD
- *PRESSURE STRESS
- *MASS FLOW RATE
- "Defining a temperature field," Section 16.11.9 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

This section describes how to specify the values of the following types of predefined fields during an analysis:

- temperature,
- field variables,
- equivalent pressure stress, and
- mass flow rate.

The procedures in which these fields can be used are outlined in "Prescribed conditions: overview," Section 32.1.1.

Temperature, field variables, equivalent pressure stress, and mass flow rate are time-dependent, predefined (not solution-dependent) fields that exist over the spatial domain of the model. They can be defined:

- by entering the data directly,
- by reading an Abaqus results file generated during a previous analysis (usually an Abaqus/Standard heat transfer analysis), or
- in an Abaqus/Standard user subroutine.

Temperature can also be defined by reading an Abaqus output database file generated during a previous analysis. In Abaqus/Standard field variables can also be defined by reading an Abaqus output database file generated during a previous analysis.

Field variables can also be made solution dependent, which allows you to introduce additional nonlinearities in the Abaqus material models.

Predefined temperature

In stress/displacement analysis the temperature difference between a predefined temperature field and any initial temperatures ("Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1) will create thermal strains if a thermal expansion coefficient is given for the material ("Thermal expansion," Section 25.1.2). The predefined temperature field also affects temperature-dependent material properties, if any. In Abaqus/Explicit temperature-dependent material properties may cause longer run times than constant properties.

You define the magnitude and time variation of temperature at the nodes, and Abaqus interpolates the temperatures to the material points.

Input File Usage: Use the following option to specify a predefined temperature field:

*TEMPERATURE

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

Restrictions

Do not specify predefined temperature fields in a pure heat transfer analysis, a coupled thermal-electrical analysis, a fully coupled temperature-displacement analysis, or a fully coupled thermal-electrical-structural analysis; instead, specify a boundary condition ("Boundary conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.3.1) to prescribe temperature degrees of freedom (11, 12, ...).

Predefined temperature fields cannot be specified in an adiabatic analysis step or in any mode-based dynamic analysis step.

To specify a predefined temperature field in a restart analysis, the corresponding predefined field must have been specified in the original analysis as either initial temperatures (see "Defining initial temperatures" in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1) or a predefined temperature field.

Predefined field variables

The usage and treatment of predefined field variables is exactly analogous to that of temperature. An example of a field variable is an electromagnetic field. Abaqus has no way of solving for such a field; rather, you can prescribe the magnitude and time variation of the field at all of the nodes of the model, and Abaqus will interpolate the values to the material points.

When prescribing field variable values, you must specify the field variable number being defined; the default is field variable number 1. Field variables must be numbered consecutively starting from one. Repeat the field variable definition to define more than one field variable.

Field variables are mainly used to change material properties depending on the field's value. For example, suppose that you wish to vary Young's modulus linearly between 30×10^6 and 35×10^6 during the response. The linear elastic material definition shown in Table 32.6.1–1 could be used.

Table 32.6.1–1 Sample material definition.

Number of field variable dependencies: 1		
Young's modulus	**Poisson's ratio**	**Value of field variable 1**
30.E6	0.3	1.0
35.E6	0.3	2.0

Define an initial condition to specify the initial value of field variable 1 as 1.0 for a node set. Then, define a predefined field variable in the analysis step to specify the value of field variable 1 as 2.0 for the node set. Young's modulus will vary smoothly over the course of the step as the field variable's value is ramped from 1.0 to 2.0 at all nodes in the node set.

Field variables can also be used to vary real properties in space by making the properties depend on field variables, as above, and by assigning different field variable values to different nodes.

Making properties depend on field variables will increase the computer time required, since Abaqus must perform the necessary table look-ups.

In an Abaqus/Standard stress/displacement analysis the difference between a predefined field variable and its initial value ("Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1) will create volumetric strains analogous to thermal strains if a field expansion coefficient (for the corresponding field variable) is given for the material ("Thermal expansion," Section 25.1.2).

Input File Usage: Use the following option to specify a predefined field variable:

*FIELD, VARIABLE=*n*

Abaqus/CAE Usage: Predefined field variables are not supported in Abaqus/CAE.

Restrictions

To specify a predefined field variable in a restart analysis, the corresponding predefined field must have been specified in the original analysis as either an initial field variable value (see "Defining initial values of predefined field variables" in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1) or a predefined field variable.

Predefined pressure stress

You can apply equivalent pressure stress as a predefined field in a mass diffusion analysis. The usage and treatment of pressure stresses is analogous to that of temperatures and field variables. In Abaqus equivalent pressure stresses are positive when they are compressive.

Input File Usage: Use the following option to specify a predefined equivalent pressure stress field:

*PRESSURE STRESS

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

Restrictions

Predefined equivalent pressure stress fields can be specified only in a mass diffusion procedure (see "Mass diffusion analysis," Section 6.9.1).

To specify a predefined equivalent pressure stress field in a restart analysis, the corresponding predefined field must have been specified in the original analysis as either initial pressure stresses (see "Defining initial pressure stress in a mass diffusion analysis" in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1) or a predefined equivalent pressure stress field.

Predefined mass flow rate

You can specify the mass flow rate per unit area (or through the entire section for one-dimensional elements) for forced convection/diffusion elements in a heat transfer analysis. The usage and treatment of mass flow rate is analogous to that of temperatures and field variables.

Input File Usage: Use the following option to specify a predefined mass flow rate field:

*MASS FLOW RATE

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

Restrictions

A predefined mass flow rate field can be specified only with forced convection/diffusion elements in a heat transfer procedure (see "Uncoupled heat transfer analysis," Section 6.5.2).

To specify a predefined mass flow rate field in a restart analysis, the corresponding predefined field must have been specified in the original analysis by using either initial mass flow rates (see "Defining initial mass flow rates in forced convection heat transfer elements" in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1) or a predefined mass flow rate field.

Reading initial values of a field from a user-specified results file

An Abaqus/Standard results file can be used to specify initial values of

- temperature (see "Defining initial temperatures" in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1);
- field variables (see "Defining initial values of predefined field variables" in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1); and
- pressure stress (see "Defining initial pressure stress in a mass diffusion analysis" in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1).

Field variable values must be read from the temperature record (see "Reading field values from a user-specified results file" below). The part (`.prt`) file from the original analysis is also required when reading data from the results file.

If the zero increment results were requested as output to the Abaqus/Standard results file (see "Obtaining results at the beginning of a step" in "Output," Section 4.1.1), you can define initial values of prescribed fields as those existing at the beginning of a step (the zero increment) in the previous heat

transfer analysis (field variables and temperatures) or stress/displacement analysis (pressure stress). The **.fil** file extension is optional.

Reading initial values of a temperature field from a user-specified output database file

An Abaqus/Standard output database file can be used to specify initial values of temperature (see "Defining initial temperatures" in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1). The part (**.prt**) file from the original analysis is also required when reading data from the output database file. Temperature values can be read between dissimilar meshes, as described in "Interpolating initial temperatures for dissimilar meshes from a user-specified results or output database file" in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1.

Initializing predefined field variables from a user-specified output database file in Abaqus/Standard

In Abaqus/Standard nodal values of temperature (NT), normalized concentrations (NNC), and electric potential (EPOT) can be used to initialize predefined fields (see "Defining initial values of predefined field variables" in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1). The part (**.prt**) file from the original analysis is also required when reading data from the output database file. The scalar nodal values can be mapped between dissimilar meshes, as described in "Defining initial predefined field variables by interpolating scalar nodal output variables for dissimilar meshes from a user-specified output database file" in "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1.

Defining time-dependent fields

The prescribed magnitude of a field can vary with time during a step according to an amplitude function. See "Prescribed conditions: overview," Section 32.1.1, and "Amplitude curves," Section 32.1.2, for details.

Input File Usage: Use one of the following options:

∗TEMPERATURE, AMPLITUDE=*amplitude_name*
∗FIELD, AMPLITUDE=*amplitude_name*
∗PRESSURE STRESS, AMPLITUDE=*amplitude_name*
∗MASS FLOW RATE, AMPLITUDE=*amplitude_name*

Abaqus/CAE Usage: In Abaqus/CAE only predefined temperature fields are available.

Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**: select region: **Distribution: Direct specification** or select an analytical field or a discrete field, **Amplitude:** *amplitude_name*

Field propagation

By default, all fields defined in the previous general analysis step remain unchanged in the subsequent general step or in subsequent consecutive linear perturbation steps. Fields do not propagate between linear perturbation steps. You define the fields in effect for a given step relative to the preexisting fields.

At each new step the existing fields can be modified and additional fields can be specified. If you specify additional values for a field, the definition of the field will be extended to those nodes where it was previously undefined. Alternatively, you can release all previously applied fields of a given type in a step and specify new ones. In this case any fields of that type that are to be retained must be respecified.

Modifying fields

By default, when you modify existing temperatures, field variables, pressure stresses, or mass flow rates, all existing values of the field remain.

Input File Usage: Use one of the following options to modify an existing field or to specify an additional field:

*TEMPERATURE, OP=MOD
*FIELD, OP=MOD
*PRESSURE STRESS, OP=MOD
*MASS FLOW RATE, OP=MOD

Abaqus/CAE Usage: In Abaqus/CAE only predefined temperature fields are available.

Load module: **Create Predefined Field** or **Predefined Field Manager**: **Edit**

Removing fields

A field that is removed is reset to the value given as an initial condition or to zero if no initial condition was defined. When fields are reset to their initial conditions, the amplitude referred to in the field definition does not apply. In Abaqus/Standard the amplitude variation defined for the step governs the behavior; in most Abaqus/Standard procedures the default is to ramp the fields back to their initial conditions (see "Procedures: overview," Section 6.1.1). In Abaqus/Explicit the values are always ramped linearly over the step back to their initial conditions.

If the temperatures, field variables, pressure stresses, or mass flow rates are reset to a new value (not to their initial conditions), the amplitude referred to in the field definition applies.

If you choose to remove any field in a step, no fields of that type will be propagated from the previous general step. All fields of the same type that are in effect during this step must be respecified.

Input File Usage: Use one of the following options to release all previously applied fields of a particular type and to specify new fields:

*TEMPERATURE, OP=NEW
*FIELD, OP=NEW
*PRESSURE STRESS, OP=NEW
*MASS FLOW RATE, OP=NEW

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

Abaqus/CAE Usage: Use the following option to reset a temperature field to the value prescribed in the initial step (or to zero if no initial value was defined):

Load module: temperature field editor: **Reset to initial**

Reading the values of a field directly from an alternate input file

The data for predefined temperature, field variables, pressure stress, or mass flow rate can be contained in a separate input file (see "Input syntax rules," Section 1.2.1).

Input File Usage: Use one of the following options:

∗TEMPERATURE, INPUT=*file_name*
∗FIELD, INPUT=*file_name*
∗PRESSURE STRESS, INPUT=*file_name*
∗MASS FLOW RATE, INPUT=*file_name*

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

Abaqus/CAE Usage: You cannot read field data from a separate input file in Abaqus/CAE.

Reading the values of a field from a user-specified file

Nodal temperatures calculated during an Abaqus/Standard heat transfer or coupled thermal-electrical analysis can be used to define temperatures in a subsequent analysis. The temperatures must have been written to the results or output database file.

If nodal temperatures are written to the results file during an Abaqus/Standard heat transfer or coupled thermal-electrical analysis, they can be used to define field variables in a subsequent analysis.

In Abaqus/Standard if nodal values of temperature (NT), normalized concentrations (NNC), or electric potential (EPOT) are written to the output database file, they can be used to define field variables in a subsequent Abaqus/Standard analysis.

In Abaqus/Standard equivalent pressure stresses calculated during a mechanical analysis can be used in a subsequent mass diffusion analysis if the element output variable SINV was written to the results file averaged at the nodes (see "Element output" in "Output to the data and results files," Section 4.1.2).

Once the data are available in a results file or output database file, they can be read into a subsequent analysis as a predefined field. Data for field variables and pressure stress can be read from a previously generated results file. In Abaqus/Standard data can also be read from a previously generated output database file. Data for temperatures can be read from a previously generated results or output database file. Data for temperatures (and field variables in Abaqus/Standard) to be interpolated between dissimilar meshes can be read only from the output database file. The part (`.prt`) file from the original analysis is also required when reading data from the results or output database file.

When the output file of an Abaqus analysis involving beam and/or shell elements is used to define temperatures, you must ensure that the number of temperature points through the section defined for corresponding elements is consistent between the two analyses. Inconsistent temperature point definition will result in an incorrect transfer of prescribed field quantities.

Reading field values from a user-specified results file

To read field values from a user-specified results file, the data must have been written to the results file as nodal output (see "Node output" in "Output to the data and results files," Section 4.1.2). Only nodal

quantities can be read from the results file. Since field variables can be written to the results file only as element quantities (record key 9), they cannot be read directly into a subsequent analysis. In this case you must generate a results file with the field data in the temperature record, even if the field variable in the current analysis is the same as a field variable in the previous analysis. Multiple results files must be generated for multiple field variables.

To generate the results file, you can write a program to create a results file (without running an Abaqus analysis) according to the format described in Chapter 5, "File Output Format." Examples of such programs are shown in that chapter. If the values will be read in as temperatures or field variables, the data must be written as nodal quantities with record key 201. If the values will be read in as a pressure stress field, the data must be averaged at the nodes (as explained in "Output to the data and results files," Section 4.1.2) and written as record key 12.

Specifying the results file to be read

You must specify the name of the results file from which the data are to be read for a temperature, field variable, or pressure stress. The **`.fil`** file extension is optional. If both **`.fil`** and **`.odb`** files exist for a temperature field and no extension is specified, the results file will be used.

Input File Usage: *TEMPERATURE, FILE=*file*
*FIELD, FILE=*file*
*PRESSURE STRESS, FILE=*file*

Abaqus/CAE Usage: In Abaqus/CAE only predefined temperature fields are available.

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

Creating a cyclic temperature history

In a direct cyclic analysis in Abaqus/Standard the temperature values must be cyclic over the step: the start value must be equal to the end value. To create a cyclic temperature history from a prior heat transfer analysis that is not cyclic, you can set the starting time, f (measured relative to the total step time period, t^σ), after which the temperatures read from the results file will be ramped back to their initial condition values. At any time point $t \geq ft^\sigma$, the temperature value is equal to

$$pTemp^\theta + (1-p)Temp^{ini},$$

where $p = \frac{(t^\sigma - t)}{(t^\sigma - ft^\sigma)}$, $Temp^{ini}$ is the initial condition value, and $Temp^\theta$ is the interpolated value obtained from the results file at time t, as illustrated in Figure 32.6.1–1.

Input File Usage: Use the following option to set the starting time for a cyclic temperature history:

*TEMPERATURE, FILE=*file*, BTRAMP=f

Abaqus/CAE Usage: Cyclic temperature histories are not supported in Abaqus/CAE.

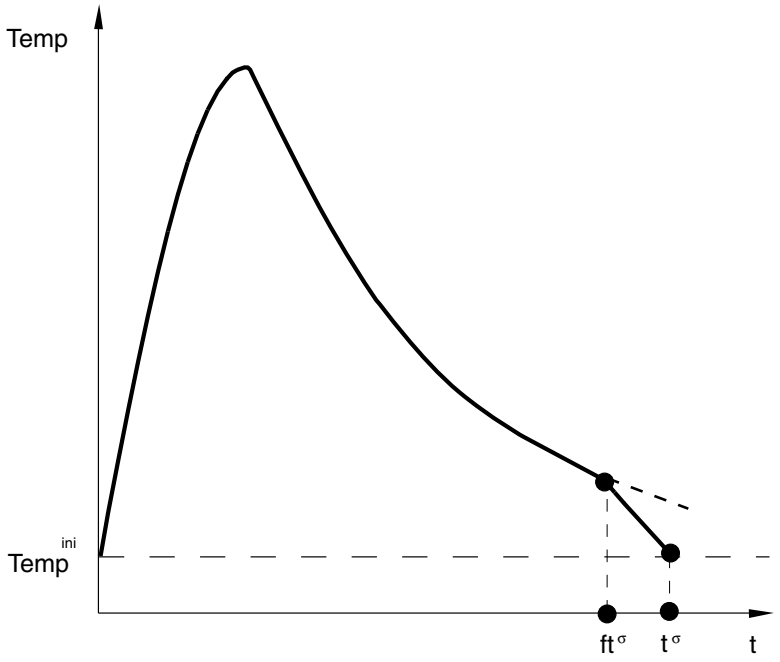


Figure 32.6.1–1 Ramp temperatures to their initial condition values after $t = ft^{\sigma}$ to create a cyclic temperature history.

Reading temperature values from a user-specified output database file

To read temperature values from a user-specified output database file, the temperatures must have been written to the output database file as nodal output (see "Node output" in "Output to the output database," Section 4.1.3).

Specifying the output database file to be read for a temperature field

You must specify the name of the output database file from which the data are to be read for a temperature field. The `.odb` extension must be included if both results and output database files exist.

Input File Usage: *TEMPERATURE, FILE=*file*

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

Defining fields using nodal scalar output values from a user-specified output database file

In Abaqus/Standard if nodal values of temperature (NT), normalized concentrations (NNC), or electric potential (EPOT) are written to the output database file, they can be used to define field variables in a

subsequent Abaqus/Standard analysis. To read these values from a user-specified output database file, they must have been written to the output database file as nodal output (see "Node output" in "Output to the output database," Section 4.1.3).

Specifying the output database file to be read for a field variable

You must specify the name of the output database file from which the data are to be read for a field variable. The **.odb** extension must be included if both results and output database files exist.

Input File Usage: *FIELD, FILE=*file*, OUTPUT VARIABLE=*scalar nodal output variable*,

Abaqus/CAE Usage: Predefined field variables are not supported in Abaqus/CAE.

Interpolating data between meshes

Data can be mapped between the same meshes, between meshes that differ only in the element order (first-order element in heat transfer analysis and second-order element in thermal-stress analysis), or between dissimilar meshes of matching element dimensionality (solid element to solid element or shell element to shell element). If data are mapped between the same meshes, no additional computations are required. To transfer data between meshes that differ only in the element order, you must activate the midside node capability. To map data between dissimilar meshes, you must activate the general interpolation capability. The midside node capability is available only for temperatures. The midside node capability and the general interpolation capability are mutually exclusive.

Using second-order stress elements with first-order heat transfer elements (the midside node capability)

In some cases it makes sense to perform an Abaqus/Standard heat transfer analysis using first-order elements followed by a thermal-stress analysis using second-order elements (and an otherwise similar mesh). For example, a heat transfer analysis including latent heat effects—for which first-order elements are best suited—can be followed by a stress analysis using second-order elements, which generally have superior deformation characteristics. In addition, the first-order temperature field calculated in the heat transfer analysis is consistent with the first-order thermal strain field provided by the second-order stress/displacement elements.

For the instances in which there is a change in the order of interpolation of element temperature variables between the heat transfer analysis and the stress analysis, temperatures must be assigned to the midside nodes of the stress/displacement elements based on the temperatures of the corner nodes of the heat transfer elements. If you specify that the midside node temperatures are needed, Abaqus will interpolate the temperatures of the midside nodes of the second-order stress/displacement elements from the corner nodes using first-order interpolation. If the midside node capability is activated in cases where both the heat transfer analysis and the stress analysis are performed with second-order elements, it is ignored. One exception is that if variable-node second-order stress/displacement elements are used in the stress analysis, activating the midside node capability will cause Abaqus to interpolate the temperatures of the midface nodes in the variable node elements from the corner or midside nodes using first-order interpolation.

Since it is assumed that the corner node temperatures have been generated in a previous heat transfer analysis, the midside node capability can be used only when the temperature field values are read from

a user-specified results or output database file. You must ensure that the nodal temperatures calculated during the heat transfer analysis are written to the results or output database file. Once the temperatures of the corner nodes are read in the subsequent stress/displacement analysis, Abaqus interpolates the midside node temperatures so that all nodes have temperatures assigned to them.

You must ensure that all temperatures of the corner nodes belonging to elements for which midside node temperatures are to be interpolated are read from the heat transfer analysis results or output database file. If the corner node temperatures are defined using a mixture of direct data input, reading from the results file or output database file, and user subroutine **UTEMP**, midside node temperatures that give unrealistic temperature fields may result. In practice, the capability for calculating midside node temperatures is most useful when temperatures generated by a heat transfer analysis are read from the results or output database file for the whole mesh during the stress analysis. Once the midside node capability is activated in a step, the capability will remain active throughout the remainder of the analysis.

Values of temperature for nodes that existed in the original analysis but do not exist in the current analysis will be ignored. Similarly, if additional nodes (but not midside nodes) exist in the current analysis, the values of fields at these nodes cannot be prescribed by reading the output files.

Input File Usage: Use the following option to interpolate temperatures between meshes that differ only in the element order:

*TEMPERATURE, FILE=*file*, MIDSIDE

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

Interpolating temperatures between dissimilar meshes (the general interpolation capability)

In some cases the model for a heat transfer analysis and the model for a thermal-stress analysis may require different meshes; for example, you may want to model a smooth temperature distribution in the heat transfer analysis and stress concentration regions in the thermal-stress analysis. Both meshes have to be different and independent of each other in such cases. Abaqus offers a general interpolation capability that allows for the use of dissimilar meshes for heat transfer and thermal-stress analyses.

The interpolation is always based on the initial (undeformed) configurations. If the mesh for which the temperature field is obtained is quite different from the initial (undeformed) configuration for the thermal-stress analysis, the interpolation may not work properly even when using the tolerance parameters discussed below.

Temperatures can be interpolated between dissimilar meshes only when the temperatures are read from an output database file. If temperatures for nodes in the heat transfer analysis that are needed for interpolation are not written to the output database file, the values at those nodes are assumed to be zero, which may lead to incorrect results for the temperature values in the stress analysis. Similarly, if additional nodes exist in the mesh for the stress analysis, the values of temperatures at these nodes are assumed to be zero. Interpolation of temperatures can also be used for specifying temperature as a

field variable in a submodel thermal-stress analysis where the temperature values are read directly from a global heat transfer analysis.

You can specify an interpolation tolerance for use in locating the nodes in the heat transfer analysis. The tolerance can be specified as an absolute value or as a fraction of the average element size. In a multistep thermal-stress analysis in which several steps read the temperature values from the same file, Abaqus interpolates the temperature values only once. If different interpolation tolerance values are used for each step, the interpolation is based on the largest specified tolerance value. If a restart analysis is performed from a particular step in the thermal-stress analysis, the restart interpolation is based on the tolerance value specified for that step.

Input File Usage: Use the following option to interpolate temperatures between dissimilar meshes:

∗TEMPERATURE, FILE=*file*`.odb`, INTERPOLATE

Use the following option to specify the interpolation tolerance as an absolute value:

∗TEMPERATURE, FILE=*file*`.odb`, INTERPOLATE, ABSOLUTE EXTERIOR TOLERANCE=*tolerance*

Use the following option to specify the interpolation tolerance as a fraction of the average element size:

∗TEMPERATURE, FILE=*file*`.odb`, INTERPOLATE, EXTERIOR TOLERANCE=*tolerance*

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

Interpolating temperatures between dissimilar meshes with user-specified regions

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

Input File Usage: Use the following option to interpolate temperatures between dissimilar meshes with user-specified regions:

∗TEMPERATURE, FILE=*file*`.odb`, INTERPOLATE, DRIVING ELSETS

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

Interpolating scalar nodal output variables between dissimilar meshes (the general interpolation capability) onto field variables in Abaqus/Standard

Abaqus/Standard offers a general interpolation capability that allows for nodal values of temperature, normalized concentration, and electric potential from one analysis to be mapped onto field variables in a subsequent analysis in the cases where the meshes in the two analyses are dissimilar.

The interpolation is always based on the initial (undeformed) configurations. If the mesh for which the field variable is obtained is quite different from the initial (undeformed) configuration for the original analysis, the interpolation may not work properly even when using the tolerance parameters discussed below.

Temperatures, normalized concentrations, and electric potentials can be interpolated between dissimilar meshes onto field variables only when they are read from an output database file. If scalar values for nodes in the current analysis that are needed for interpolation are not written to the output database file, the values at those nodes are assumed to be zero, which may lead to incorrect results for the field variables. Similarly, if additional nodes exist in the mesh for the current analysis, the values of the field variables at these nodes are assumed to be zero.

You can specify an interpolation tolerance for use in locating the nodes in the original analysis. The tolerance can be specified as an absolute value or as a fraction of the average element size. In a multistep analysis in which several steps read nodal output variables values from the same file, Abaqus interpolates the nodal values only once. If different interpolation tolerance values are used for each step, the interpolation is based on the largest specified tolerance value. If a restart analysis is performed from a particular step in the original analysis, the restart interpolation is based on the tolerance value specified for that step.

Input File Usage: Use the following option to interpolate scalar nodal output variables between dissimilar meshes:

∗FIELD, FILE=*file*`.odb`, OUTPUT VARIABLE=*scalar nodal output variable*, INTERPOLATE

Use the following option to specify the interpolation tolerance as an absolute value:

∗FIELD, FILE=*file*`.odb`, OUTPUT VARIABLE=*scalar nodal output variable*, INTERPOLATE, ABSOLUTE EXTERIOR TOLERANCE=*tolerance*

Use the following option to specify the interpolation tolerance as a fraction of the average element size:

∗FIELD, FILE=*file*`.odb`, OUTPUT VARIABLE=*scalar nodal output variable*, INTERPOLATE, EXTERIOR TOLERANCE=*tolerance*

Abaqus/CAE Usage: Predefined field variables are not supported in Abaqus/CAE.

Specifying the step and increment to be read from the file

You can specify the first and last step, respectively, from which results will be read. Similarly, you can specify the first and last increment, respectively, from which results will be read. You can specify any combination of these values. Any zero-increment file output that is present in the results file of an Abaqus/Standard analysis (written only if the zero increment results are requested; see "Obtaining results at the beginning of a step" in "Output," Section 4.1.1) will be ignored. Results must have been written to the results or output database file at the specified step and increment.

If you do not specify the first step from which to read, Abaqus will begin reading results from the first step available in the results or output database file.

If you do not specify the first increment from which to read, Abaqus will begin reading results from the first increment available in the first step from which results will be read (the first increment following the zero increment if zero-increment file output is present in the results file).

If you do not specify the last step from which to read, the first step from which results will be read will also be the last step.

If you do not specify the last increment from which to read, Abaqus will read the results or output database file until it reaches the last available increment in the last step from which results will be read.

Input File Usage: Use one of the following options:

∗TEMPERATURE, FILE=*file*, BSTEP=*bstep*, BINC=*binc*, ESTEP=*estep*, EINC=*einc*
∗FIELD, FILE=*file*, BSTEP=*bstep*, BINC=*binc*, ESTEP=*estep*, EINC=*einc*
∗PRESSURE STRESS, FILE=*file*, BSTEP=*bstep*, BINC=*binc*, ESTEP=*estep*, EINC=*einc*

For example, the following input would read temperature data from output database file **heat.odb** beginning at Step 2, increment 2, and ending at Step 3, increment 5:

```
*TEMPERATURE, FILE=heat.odb, BSTEP=2, BINC=2,
ESTEP=3, EINC=5
```

Abaqus/CAE Usage: In Abaqus/CAE only predefined temperature fields are available.

Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**: select region: **Distribution: From results or output database file**, **File name:** *file*, **Begin step:** *bstep*, **Begin increment:** *binc*, **End step:** *estep*, and **End increment:** *einc*

Interpolation in time

When Abaqus reads temperature, field variable, or equivalent pressure stress data from a results file or temperatures from an output database file, it must obtain values of the field at the time points used by the analysis. Since data corresponding to these time points are usually not present in the results or output database files, Abaqus will interpolate linearly in time between the time points stored in the file to obtain

values at the time points required by the analysis. Since the interpolation is linear, you must take care to provide sufficient data in the results or output database file to make this interpolation meaningful.

For the purpose of such interpolation the time period of the results being read in is determined as follows:

- The period starts at the time of the most recent increment written, of the relevant field, that precedes the beginning increment (either user-specified or default). For example if your results file contains temperature field data at increments 5, 10, and 15; and you specify a beginning increment number of 10 when reading these results; the results period starts with the time associated with increment 5 since that is the most recent increment that precedes the specified beginning increment of 10. You can ensure that the results starting time matches the beginning time of the beginning increment you specify by writing the results data with an increment frequency of 1.
- The period ends at the completion of the ending increment (either user-specified or default).

If the analysis requires data at a time point prior to the first increment for which data are available in the either of files, Abaqus will interpolate between the given initial condition data and the data of the first increment stored in the file.

Reading results for multiple fields

If data for multiple fields are being read in the same step and the time values corresponding to the starting step and increment or to the ending step and increment are different for different fields, Abaqus interpolates through the total time period from the earliest time point chosen in any file to the latest. For example, suppose the starting increment in the starting step in the temperature file begins at 3 sec and the ending increment in the ending step ends at 6 sec. During the same step we also read field variable data, for which the starting increment in the starting step begins at 2 sec and the ending increment in the ending step ends at 5 sec. In such a case the time period used for interpolation is from 2 sec to 6 sec.

Automatic adjustment of the time scale

It is convenient to set the period of the step equal to the time period of the files being read in. Otherwise, Abaqus will automatically scale the time period from the results or output database file to match the time period of the stress analysis. The scale factor is t^σ/t^θ, where t^σ is the time period of the stress analysis and t^θ is the total time period obtained from all results or output database files, as described above.

Obtaining results at a particular point in time

In Abaqus/Standard it is sometimes desirable to carry out a calculation corresponding to the field values at a particular point in time. For example, suppose that temperature data are available in the output file for increment 10 at time $t = 4$ and increment 15 at time $t = 5$ and that you wish to carry out a static analysis based on temperature values at $t = 4.5$. In this case Abaqus must interpolate linearly between the results at $t = 4$ and $t = 5$ to obtain the intermediate result at $t = 4.5$. To accomplish this task, you should specify an initial time increment of 4.5 and a time period of 5. for the static analysis step and read the temperature values from the output file starting at Step 1, Increment 1 and ending at Step 1, Increment 15. Specifying a starting increment of 1 instead of 10 ensures that t^θ is the entire time period stored in

the output file, not just the period between increments 10 and 15; hence, the scale factor between the output file data and the static analysis is unity, and the initial time of 4.5 has the desired meaning.

Initial transients

To track initial transients accurately, Abaqus/Standard may automatically reduce the initial time increment for the step. If the user-specified suggested initial time increment is greater than the scaled value of the first time increment read from the Abaqus/Standard results file, Abaqus/Standard will use that scaled value.

Restrictions

The following restrictions exist:

- Temperatures and field variables cannot be read from a user-specified file in a modified Riks static analysis step ("Unstable collapse and postbuckling analysis," Section 6.2.4).
- Temperature cannot be interpolated from a coupled thermal-electrical analysis.
- Equivalent pressure stress cannot be read from the results file if the model is defined in terms of an assembly of part instances.
- In Abaqus/Explicit field variables cannot be read from the output database file.
- Pressure stress cannot be read from the output database file.
- Elements that do not support interpolation for temperature mapping include the complete libraries of convective heat transfer elements, axisymmetric elements with nonlinear axisymmetric deformation, axisymmetric surface elements, hydrostatic fluid elements, solid infinite stress elements, and coupled thermal/electrical elements. Other specific elements that are not supported include: GKPS6, GKPE6, GKAX6, GK3D18, GK3D12M, GK3D4L, GK3D6L, GKPS4N, GKAX6N, GK3D18N, GK3D12MN, GK3D4LN, and GK3D6LN.

Defining the values of a predefined field in a user subroutine

In Abaqus/Standard you can specify predefined temperatures, field variables, equivalent pressure stresses, or mass flow rates at the nodes in a user subroutine. Temperature values can be defined in user subroutine **UTEMP**; field variable values, in user subroutine **UFIELD**; equivalent pressure stress values, in user subroutine **UPRESS**; and mass flow rates, in user subroutine **UMASFL**.

The user subroutine (**UTEMP**, **UFIELD**, **UPRESS**, or **UMASFL**) will be called for each specified node. Field values entered directly will be ignored. If a results or output database file has been specified in addition to the user subroutine, values read from the results or output database file will be passed into the user subroutine for possible modification.

Input File Usage: Use one of the following options:

```
*TEMPERATURE, USER
*FIELD, USER
*PRESSURE STRESS, USER
*MASS FLOW RATE, USER
```

Abaqus/CAE Usage: In Abaqus/CAE only predefined temperature fields are available.

Load module: **Create Predefined Field**: **Step:** *analysis_step*: choose **Other** for the **Category** and **Temperature** for the **Types for Selected Step**: select region: **Distribution: User-defined** or **From results or output database file and user-defined**

Updating multiple predefined field variables

If multiple field variables are predefined, only one field variable at a time can be redefined in user subroutine **UFIELD**. There are situations in which the analysis requires a number of field variables that are predefined with respect to the solution but depend on each other. You can specify the number of field variables to be updated simultaneously at a point, *n*. Abaqus/Standard passes information about *n* field variables at each specified node into **UFIELD**.

You can update all or part of the field variables used in the analysis but must remember that the field variables are numbered consecutively from 1. If, for example, you have four field variables in the analysis and want to update the second and third variables simultaneously in subroutine **UFIELD**, you must specify *n*=3. In this case Abaqus/Standard passes information about the first three field variables into subroutine **UFIELD**, and you update only the second and third variables.

Input File Usage: *FIELD, USER, NUMBER=*n*

Abaqus/CAE Usage: Predefined field variables are not supported in Abaqus/CAE.

Defining solution-dependent field variables

In Abaqus/Standard solution-dependent field variables can be defined in user subroutine **USDFLD**. The values of predefined field variables or initial fields can be passed into user subroutine **USDFLD** and can be changed in that routine—see "Material data definition," Section 20.1.2.

Changes to the field variables in **USDFLD** are local to the material point and do not affect the nodal values.

Data hierarchy

If both results or output database file input and direct data input are used in the same step, the direct data input will take precedence if both define the field at the same node. If user subroutine input is specified, the values given directly are ignored and the user subroutine modifies the values read from the results or output database file.

Element type considerations

You can specify either one or several values of a predefined field at a node, depending on the element type that is used. You should note the following considerations when choosing the form of predefined field specification.

Use in a mass diffusion analysis

For solid elements only one value can be given at a node. Since only solid elements can be used in mass diffusion analysis, this is the only way to define equivalent pressure stresses at a node.

Use with beam and shell elements

The following possibilities exist for temperatures and field variable specification in beam and shell elements:

- For shell and beam elements with general cross-section definitions, the temperature and field variable magnitude at points in the section is defined by the value at the reference surface. Any gradient of these variables specified across the section is ignored.
- For shell and beam elements with cross-sections that require numerical integration, the temperature and field variable magnitudes at points in the section can be defined either from the value at the reference surface and the gradient or gradients across the section or by giving the values at a number of points across the section. The choice between these two methods is made in the section definition (see "Specifying temperature and field variables" in "Using a shell section integrated during the analysis to define the section behavior," Section 28.6.5, and "Specifying temperature and field variables" in "Using a beam section integrated during the analysis to define the section behavior," Section 28.3.6, for details).

See Part VI, "Elements," for the details of use with each element type. The default, if only one value is given, is a constant magnitude across the section.

Temperature and field variable compatibility across elements

Abaqus assumes that the field definitions (including initial conditions) at all the nodes of any element are compatible with the field definition method chosen for the element. Cases may arise where the definition of a field changes from one element to the next (for example, when two adjacent shell elements have a different number of section points through the thickness or when the temperature and field variable magnitudes for one beam element are defined by giving the values at a number of points across the section while those for the abutting beam element are defined from the value at the reference surface and the gradient or gradients across the section). In these cases separate nodes should be used on the interface between such elements and multi-point constraints should be applied to make the displacements and rotations the same at corresponding nodes (see "General multi-point constraints," Section 33.2.2); otherwise, the fields on the nodes at the interface will be used for each adjacent element with the field definition method chosen for the element.

Part VIII: Constraints

- Chapter 33, “Constraints”

33. Constraints

Overview 33.1
Multi-point constraints 33.2
Surface-based constraints 33.3
Embedded elements 33.4
Element end release 33.5
Overconstraint checks 33.6

33.1 Overview

- “Kinematic constraints: overview,” Section 33.1.1

33.1.1 KINEMATIC CONSTRAINTS: OVERVIEW

The following types of kinematic constraints can be defined:

- **Equations:** Linear multi-point constraints can be given in the form of an equation (see "Linear constraint equations," Section 33.2.1).
- **Multi-point constraints:** Multi-point constraints (MPCs) specify linear or nonlinear constraints between nodes. These relations between nodes can be the default types that are provided in Abaqus or, in Abaqus/Standard, can be coded in the form of a user subroutine. "General multi-point constraints," Section 33.2.2, explains the use of MPCs and lists the available default constraints.
- **Kinematic coupling:** In Abaqus/Standard a node or group of nodes can be constrained to a reference node. Similar to multi-point constraints, the kinematic coupling constraint allows general node-by-node specification of constrained degrees of freedom (see "Kinematic coupling constraints," Section 33.2.3).
- **Surface-based tie constraints:** Two surfaces can be tied together. Each node on the first surface (the slave surface) will have the same values for its degrees of freedom as the point on the second surface (the master surface) to which it is closest (see "Mesh tie constraints," Section 33.3.1). In the case of surface elements tied to a beam surface, the offset distances between the surface elements and the beam are used in the definition of constraints, which include the rotational degrees of freedom of the beam.
- **Surface-based coupling constraints:** A group of nodes located on a surface can be constrained to a reference node. This constraint may be kinematic, in which the group of coupling nodes can be constrained to the rigid body motion defined by the reference node, or distributing, in which the group of coupling nodes can be constrained to the rigid body motion defined by the reference node in an average sense (see "Coupling constraints," Section 33.3.2).
- **Surface-based shell-to-solid coupling:** An edge-based surface on a three-dimensional shell element mesh can be coupled to an element- or node-based surface on a three-dimensional solid mesh. The coupling is enforced by the creation of an internal set of distributing coupling constraints (see "Shell-to-solid coupling," Section 33.3.3).
- **Mesh-independent spot welds:** Two or more surfaces can be bonded together using fasteners such as spot welds (see "Mesh-independent fasteners," Section 33.3.4). Distributed coupling constraints are created on each of the connected surfaces. The connection is modeled independent of the mesh.
- **Embedded elements:** An element or a group of elements can be embedded in a group of host elements (see "Embedded elements," Section 33.4.1). Abaqus will search for the geometric relationships between nodes on the embedded elements and the host elements. If a node on an embedded element lies within a host element, the degrees of freedom at the node will be eliminated by constraining them to the interpolated values of the degrees of freedom of the host element. Host elements cannot be embedded themselves.
- **Release:** In Abaqus/Standard a local rotational degree of freedom or a combination of local rotational degrees of freedom can be released at one or both ends of a beam element (see "Element end release," Section 33.5.1).

Boundary conditions are also a type of kinematic constraint in stress analysis because they define the support of the structure or give fixed displacements at nodal points. Specification of boundary conditions is discussed in "Boundary conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.3.1.

Connector elements can be used to impose element-based kinematic constraints for mechanism-type analysis. See "Connectors: overview," Section 30.1.1.

Contact interactions, described in Part IX, "Interactions," can be used to enforce constraints between bodies that come into contact. Contact interactions can be used in mechanical as well as coupled thermal-mechanical, coupled thermal-electrical-structural, and coupled pore fluid-mechanical analysis.

"Overconstraint checks," Section 33.6.1, describes the overconstraint checks and the automatic resolution of some overconstraints performed in Abaqus/Standard.

Multiple kinematic constraints at a node

It is possible to use a single node in several multi-point constraints, kinematic coupling constraints, tie constraints, and constraint equations. However, the constraint dependencies are handled differently in Abaqus/Standard and Abaqus/Explicit.

Multiple constraints in Abaqus/Standard

In Abaqus/Standard kinematic constraints are usually imposed by eliminating degrees of freedom at the dependent nodes. Once a variable has been eliminated, it cannot be referenced in any boundary condition or in any subsequent multi-point constraint, kinematic coupling constraint, tie constraint, or constraint equation. If you intend to use a variable that is eliminated in one constraint equation as the retained variable in another constraint equation, you must order the input so that the constraint equation in which the variable is eliminated follows the other constraint equations. MPC types BEAM, CYCLSYM, LINK, PIN, REVOLUTE, TIE, and UNIVERSAL, as well as the kinematic coupling and tie constraints, are sorted internally by Abaqus/Standard to obtain a proper elimination order when possible.

Excessive chaining of multi-point constraints, kinematic coupling constraints, and constraint equations is not recommended and may result in a degradation in performance during analysis preprocessing. Whenever possible, it is best to relate the behavior of several nodes (grouped into a node set) to a single node by using one multi-point constraint, kinematic coupling constraint, or constraint equation.

Multiple constraints in Abaqus/Explicit

Kinematic constraints in Abaqus/Explicit can be defined in any order without regard to constraint dependencies. With the exception of constraints arising from kinematic contact pairs, Abaqus/Explicit solves for all kinematic constraints simultaneously. Thus, nodes involved in a combination of multi-point constraints, constraint equations, connector element kinematic constraints, rigid body constraints, and constraints due to boundary conditions will simultaneously satisfy these constraints as long as they are not conflicting. Redundant and closed loop constraints are acceptable.

Since the above constraints are enforced independent of contact constraints, the penalty contact algorithm should be used for nodes involved in both kinematic constraints and contact pair definitions. The penalty contact algorithm introduces numerical softening through the use of penalty springs and does

not interfere with kinematic constraints. If a node that participates in a kinematic constraint is used in a kinematic contact pair, the contact constraint will most likely override the kinematic constraint. Except for rigid bodies, Abaqus/Explicit will not prevent you from defining these conditions, but the results cannot be guaranteed. If a kinematic constraint is defined for a node on a rigid body, the penalty contact algorithm must be used for all contact pairs involving the rigid body.

To obtain accurate reaction force and moment output from Abaqus/Explicit at nodes that are constrained by boundary conditions in addition to one or more of the kinematic constraints described above, it may sometimes be necessary to run the analysis in double precision. In such a situation a double precision run will also yield a better estimate of the work done by the reaction forces and moments, thereby providing a more accurate value of the energy due to the external work reported by Abaqus/Explicit.

Abaqus/Explicit uses a penalty method to solve for constraints in certain situations. The penalties are weighted based on the masses of nodes participating in the constraint and the stable time increment. The penalty formulation attempts to satisfy the constraint approximately (i.e., a very small lack of compliance exits after imposition of the constraint). One situation in which the penalty approach is used to solve the constraint is when slave nodes of a tie constraint participate in other constraints such as multi-point constraints, kinematic coupling constraints, constraint equations, connector elements, rigid body constraints, or constraints due to boundary conditions. In this case the lack of compliance in the tie constraint is not carried across step boundaries; therefore, noisy accelerations and energy imbalance may be observed at step boundaries for certain problems. An alternative modeling approach (such as simply reversing the master and slave surfaces in the tie constraint) may switch to a different solution approach and thus resolve the above mentioned inaccuracies.

In Abaqus/Explicit when there are two or more overlapping distributed coupling constraints or overlapping distributed coupling and tie constraints, and the elements underlying the participating surfaces have very low densities, the lack of compliance may result in an inaccurate solution. Specifying reasonable density values for underlying elements may reduce the lack of compliance and improve solution accuracy.

Abaqus/Explicit always uses a geometrically nonlinear formulation for the enforcement of kinematic constraints. This is the case even when you have designated a particular analysis step as being geometrically linear. Consequently, results in these geometrically linear analyses could be hard to interpret, particularly when the loading in the model is high (displacements are large) and a geometrically nonlinear formulation should have been used.

Initial conditions at constrained nodes

You should not think of initial conditions as boundary conditions at the beginning of the analysis. When you prescribe initial conditions at a set of nodes that are constrained kinematically, Abaqus processes the prescribed values to determine an initial value that is then redistributed to the nodes involved in the constraints in a kinematically consistent manner via a “mass” weighted averaging method: the initial value prescribed at each node involved in the constraint is weighted with the corresponding “mass” at the node. Consequently, the values of the initial conditions that you specified at the nodes are recomputed, and in many cases the output of the prescribed quantity at these nodes at the beginning of the analysis will

be different from the values that you have specified. Correct modeling practices consist of specifying initial conditions at all nodes involved in the constraints in a manner consistent with constraint itself.

This behavior is probably best understood via a simple example. Consider a model consisting of two nodes each with a mass of 1.0 constrained by boundary conditions in global directions 2 and 3 and allowed to move freely along the global 1-direction while their relative motions is also constrained via a rigid connection such as a BEAM connector. Assume that you have specified an initial translational velocity along the global 1-direction only at the first node of 10.0 units and you have not specified initial conditions at the second node. Consequently, Abaqus will consider that the initial velocity is 0.0 at the second node. This initial velocity field is inconsistent with the kinematic constraint enforced by the BEAM connector because the constraint would be violated if the initial conditions were to be enforced even for an infinitesimally short period of time. The outcome is that Abaqus will compute an initial velocity field that would redistribute the momentum of the first node in a manner consistent with the constraint. In this particular example, the net effect is that both nodes will end up with an initial velocity of 5.0 units along the global 1-direction. Most likely, this is not what you intended. Correct modeling practice in this case would be to specify an initial velocity of 10.0 units at both nodes involved in the constraint. In this case Abaqus will still recompute the initial values, but the outcome would be an initial velocity of 10.0 units at both nodes, as intended.

The same principle applies in more complicated modeling situations. For example, if you prescribe initial translational velocities at the nodes of the kinematic constraint, an average translational velocity of the constrained nodes is computed by calculating a mass weighted average of the velocities at the individual nodes. Depending on the nature of the kinematic constraint, initial translational velocities at the nodes of a constraint may also give rise to an average rotational velocity about the center of mass of the constraint. The velocity of each individual node of the constraint is then recomputed from the average translational and rotational velocities at the center of mass of the constraint. The "mass"-type quantity used in the weighting varies depending on the nature of the prescribed quantity: if the initial condition is prescribed on the rotational velocities, the rotary inertia at the nodes is used in the weighting; if temperature initial conditions are prescribed, the thermal capacitance at the nodes is used in the weighting; and so on.

In all cases, you should specify initial conditions at all nodes involved in the constraint that are consistent with the constraint. This is typically accomplished by specifying the same initial conditions at all nodes involved in the constraint.

33.2 Multi-point constraints

- “Linear constraint equations,” Section 33.2.1
- “General multi-point constraints,” Section 33.2.2
- “Kinematic coupling constraints,” Section 33.2.3

33.2.1 LINEAR CONSTRAINT EQUATIONS

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Kinematic constraints: overview," Section 33.1.1
- *EQUATION
- "Defining equation constraints," Section 15.15.9 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

A linear multi-point constraint requires that a linear combination of nodal variables is equal to zero; that is, $A_1 u_i^P + A_2 u_j^Q + \cdots + A_N u_k^R = 0$, where u_i^P is a nodal variable at node ***P***, degree of freedom *i*; and the A_n are coefficients that define the relative motion of the nodes.

In Abaqus/Explicit linear constraint equations can be used only to constrain mechanical degrees of freedom.

Defining a linear constraint equation

A linear constraint equation is defined in Abaqus by specifying:

- the number of terms in the equation, ***N***;
- the nodes, ***P***, and the degrees of freedom, *i*, corresponding to the nodal variables u_i^P; and
- the coefficients, A_n.

For example, to impose the equation

$$u_3^5 = u_1^6 - u_3^{1000},$$

you would first write the equation in the standard form,

$$u_3^5 - u_1^6 + u_3^{1000} = 0.$$

There are three terms in this equation (***N***=3). ***P***=5, *i*=3, A_1=1.0, ***Q***=6, *j*=1, A_2=−1.0, ***R***=1000, *k*=3, and A_3=1.0.

Input File Usage: *EQUATION
N
P, *i*, A_1, ***Q***, *j*, A_2, *etc.*

For example, the following input could be used to define the equation constraint above:

```
*EQUATION
 3
 5, 3, 1.0, 6, 1, -1.0, 1000, 3, 1.0
```

Either node sets or individual nodes can be specified as input. If node sets are used, corresponding set entries will be matched to each other. If sorted node sets are given as input, you must ensure that the nodes are numbered such that they will match up with each other correctly once sorted. The nodes in an unsorted node set will be used in the order that they are given in defining the set (see "Node definition," Section 2.1.1).

If the first entry is a single node, subsequent entries must be single nodes. If the first entry is a node set, subsequent entries can be either node sets or single nodes. The latter option is useful if a degree of freedom at each of a set of nodes depends on a degree of freedom of a single node, such as may occur in certain symmetry conditions or in the simulation of a rigid body.

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Equation**

The nodes must be specified as sets. The first set can contain one or more points. Subsequent sets must contain only a single point.

In Abaqus/Standard the first nodal variable specified (u_i^P corresponding to A_1) will be eliminated to impose the constraint (in the above equation constraint, degree of freedom 3 at node 5 will be eliminated); therefore, it should not be used to apply boundary conditions, nor should it be used in any subsequent multi-point constraint, kinematic coupling constraint, tie constraint, or equation constraint (see "Kinematic constraints: overview," Section 33.1.1). In addition, the coefficient A_1 should not be set to zero. These restrictions do not apply in Abaqus/Explicit.

In Abaqus/Standard a linear multi-point constraint cannot be used to connect two rigid bodies at nodes other than the reference nodes, since multi-point constraints use degree-of-freedom elimination and the other nodes on a rigid body do not have independent degrees of freedom. In Abaqus/Explicit a rigid body reference node or any other node on a rigid body can be used in an equation constraint definition.

Use with transformed coordinate systems

If a local coordinate system ("Transformed coordinate systems," Section 2.1.5) is defined for any node involved in the equation, the variables at that node appear in the equation in the local system.

Use within a part

If an equation constraint is defined at the part (or part instance) level, the nodal variables are transformed initially according to the positioning data given for each instance of the part (see "Defining an assembly," Section 2.10.1).

Note: Equation constraints cannot be defined at the part (or part instance) level in Abaqus/CAE.

Prescribing a nonhomogeneous constraint

It is sometimes necessary to impose a constraint in the form

$$A_1 u_i^P + A_2 u_j^Q + \cdots + A_N u_k^R = \hat{u},$$

where $\hat{u}(t)$ is a prescribed value that may vary with time, *t*. This is easily done by rewriting the equation as

$$A_1 u_i^P + A_2 u_j^Q + \cdots + A_N u_k^R - \hat{u}_m^Z = 0$$

and introducing a node, *Z*, that is not attached to any element in the model. Choosing $\hat{u}_m^Z$ to be some convenient degree of freedom *m* at node *Z* allows the prescribed value $\hat{u}(t)$ to be imposed through a boundary condition specification. If necessary, an amplitude reference can be provided to give the variation with time (see "Boundary conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.3.1); such an amplitude reference is required in Abaqus/Explicit for prescribed displacements.

For example, assume that node 1000 in the example above is a "dummy" node that appears only in this equation and is not attached to any other part of the model. Defining a boundary condition to constrain degree of freedom 3 at node 1000 to −12.5 would impose the constraint

$$u_3^5 - u_1^6 = 12.5.$$

Constraint forces and global equilibrium

Linear constraint equations introduce constraint forces at all degrees of freedom appearing in the equations. These forces are considered external, but they are not included in reaction force output. Therefore, the totals provided at the end of the reaction force output tables may reflect an incomplete measure of global equilibrium.

To illustrate this behavior, consider a spring-supported beam subjected to a concentrated load as shown in Figure 33.2.1–1. The static reaction forces are $R_y^C = -3$ and $R_y^D = -6$. In Figure 33.2.1–2 the same structure is subjected to the additional linear constraint equation $u_y^A - u_y^B = 0$, which constrains the beam to remain horizontal. This introduces constraint forces $F_y^A = 1.5$ and $F_y^B = -1.5$, and the new reaction forces are $R_y^C = R_y^D = -4.5$. These reaction forces produce a global force balance in the Y-direction, but since the constraint forces are not included in reaction force output, the global moment balance about point A cannot be verified.

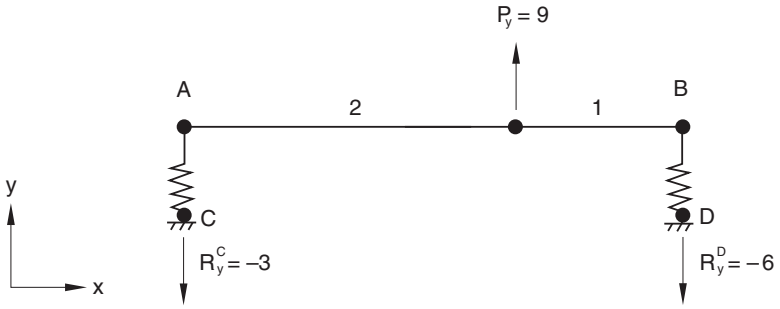


Figure 33.2.1–1 Beam with no linear constraints.

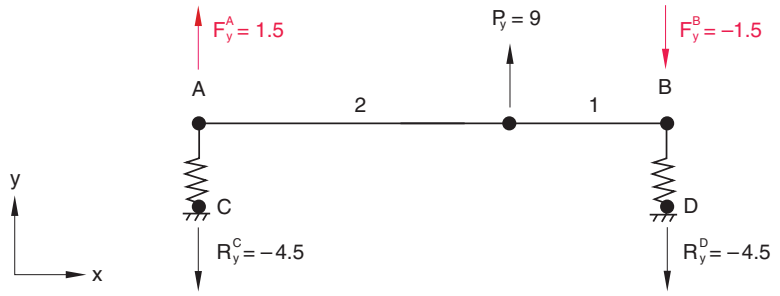


Figure 33.2.1–2 Beam with linear constraint $u_y^A - u_y^B = 0$. Constraint forces F_y^A and F_y^B are not included in reaction force output.

The global force balance can also be incomplete. This is demonstrated in Figure 33.2.1–3, where a pulley connection between nodes A and B is represented by the linear constraint equation $u_y^A - u_x^B = 0$. The constraint forces at the pulley, F_x and F_y, are not included in the reaction force output, producing incomplete global force balances in both the *X*- and *Y*-directions.

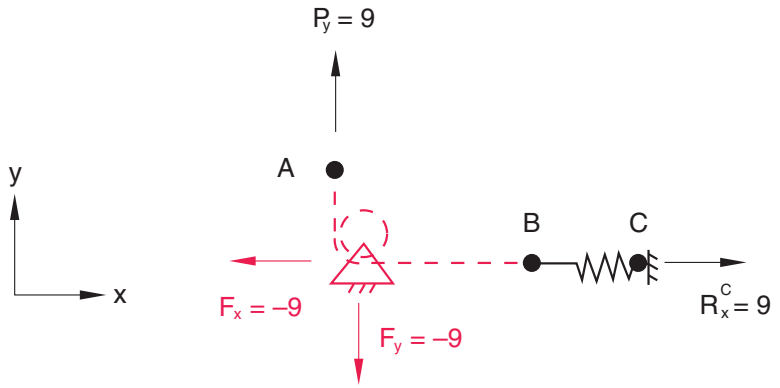


Figure 33.2.1–3 Pulley connection represented by the linear constraint $u_y^A - u_x^B = 0$. Constraint forces F_x and F_y are not included in reaction force output.

Obtaining the constraint force

The linear constraint generates constraint forces at all the degrees of freedom involved in the equation. For a given constraint equation these forces are proportional to their respective coefficients. To find the constraint forces, introduce a node ***Z*** that is not attached to any element in the model; rewrite the constraint equation as

$$A_1 u_i^P + A_2 u_j^Q + \cdots + A_N u_k^R - A_1 \hat{u}_m^Z = 0;$$

and specify a zero displacement boundary condition at degree of freedom ***m*** of node ***Z***. The reaction force obtained at node ***Z*** will be equal to the constraint force acting at node ***P*** in degree of freedom ***i***. The constraint force in any term with coefficient A_K in the constraint equation is obtained by multiplying the constraint force at node ***P*** in degree of freedom ***i*** with the ratio A_K/A_1. For example, if the equation is

$$u_3^5 - u_3^6 = 0$$

and the forces in the constraint are needed, the equation can be rewritten as

$$u_3^5 - u_3^6 - u_3^{1000} = 0,$$

where node 1000 is the fixed "dummy" node. Since the coefficient of u_3^5 is the opposite of the coefficient of u_3^{1000}, the constraint force at node 5 is the same as the reaction force at node 1000. Since the coefficient of u_3^6 is the same as the coefficient of u_3^{1000}, the constraint force at node 6 is the opposite of the reaction force at node 1000.

Defining a constraint in a deformed state

Sometimes we may wish to impose an equation starting at a certain point in the analysis:

$$A_1 \Delta u_i^P + A_2 \Delta u_j^Q + \ldots + A_N \Delta u_k^R = 0,$$

where Δu represents the change in displacement after time t_0. The equation can be rewritten as

$$A_1 u_i^P + A_2 u_j^Q + \ldots + A_N u_k^R - \hat{u}_m^Z = 0,$$

where, again, node Z is not attached to any element in the model. Prior to time t_0 (which is assumed to be at the end of a step), degree of freedom m of node Z is left unrestrained. After time t_0 further changes in $\hat{u}_m^Z$ are restrained in Abaqus/Standard by applying a boundary condition fixing the degree of freedom at its current values at the start of the step.

Reading the data from an alternate input file

The input for a linear constraint equation can be contained in a separate input file.

Input File Usage: ∗EQUATION, INPUT=*file_name*

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

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Equation**: click mouse button 3 while holding the cursor over the data table, and select **Read from File**

33.2.2 GENERAL MULTI-POINT CONSTRAINTS

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Kinematic constraints: overview," Section 33.1.1
- *MPC
- "Defining MPC constraints," Section 15.15.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- Chapter 24, "Connectors," of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Multi-point constraints (MPCs):

- allow constraints to be imposed between different degrees of freedom of the model; and
- can be quite general (nonlinear and nonhomogeneous).

The most commonly required constraints are available directly by choosing an MPC type and giving the associated data. The available MPC types are described below; MPCs that are available only in Abaqus/Standard are designated with an (S).

In Abaqus/Standard the constraints can also be given by user subroutine **MPC**.

Linear constraints can be given directly by defining a linear constraint equation (see "Linear constraint equations," Section 33.2.1).

In Abaqus/Explicit some multi-point constraints can be modeled more effectively using rigid bodies (see "Rigid body definition," Section 2.4.1).

Several MPC types are also available with connector elements ("Connector elements," Section 30.1.2). Although the connector elements impose the same kinematic constraint, connectors do not eliminate degrees of freedom.

MPC constraint forces are not available as output quantities. Therefore, to output the forces required to enforce the constraint specified in an MPC, you should use an equivalent connector element. Connector element force, moment, and kinematic output is readily available and is defined in "Connector element library," Section 30.1.4.

Identifying the nodes involved in the MPC

For any MPC type, either node sets or individual nodes can be given as input. If the first entry is a node, subsequent entries must be nodes. If the first entry is a node set, subsequent entries can be either node sets or single nodes. The latter option is useful if a degree of freedom at each of a set of nodes depends on a degree of freedom of a single node, such as may occur in certain symmetry conditions or in the simulation of a rigid body.

If node sets are used, corresponding set entries will be constrained to each other. If sorted node sets are given as input, you must ensure that the nodes are numbered such that they will match up correctly when sorted. The nodes in an unsorted node set (see "Node definition," Section 2.1.1) will be used in the order that they are given in defining the set.

In Abaqus/Standard multi-point constraints cannot be used to connect two rigid bodies at nodes other than the reference nodes, since multi-point constraints use degree-of-freedom elimination and the other nodes on a rigid body do not have independent degrees of freedom. In Abaqus/Explicit a rigid body reference node or any other node on a rigid body can be used in a multi-point constraint definition.

Abaqus/CAE uses connectors to define multi-point constraints between two points and constraints to define multi-point constraints between a point and slave nodes in a region. Set-to-set multi-point constraints and unsorted node sets are not supported in Abaqus/CAE.

Input File Usage: *MPC

Abaqus/CAE Usage: Use the following options to define a multi-point constraint between two points:

Interaction module:
Connector→Geometry→Create Wire Feature
Connector→Section→Create: **Connection Category**: **MPC**,
MPC type: select type
Connector→Assignment→Create: select wires: **Section**:
select MPC connector section

Use the following options to define a multi-point constraint between a point and slave nodes in a region:

Interaction module:
Constraint→Create: **MPC Constraint**: select control point and region; **MPC type**: select type

Use with transformed coordinate systems

Local coordinate systems (see "Transformed coordinate systems," Section 2.1.5) can be defined for any nodes connected to MPCs. Some special considerations apply for user-defined MPCs, as described in "MPC," Section 1.1.14 of the Abaqus User Subroutines Reference Manual.

Defining multiple multi-point constraints at a point

See "Kinematic constraints: overview," Section 33.1.1, for details on how multiple kinematic constraints at a point are treated in Abaqus/Standard and Abaqus/Explicit.

In Abaqus/Standard MPCs are usually imposed by eliminating the degree of freedom at the first node given (the dependent degree of freedom). MPC types BEAM, CYCLSYM, LINK, PIN, REVOLUTE, TIE, and UNIVERSAL are sorted internally by Abaqus/Standard so that the MPC in which a node is used as a dependent node is the last MPC that uses this node. Therefore, groups of these MPCs can be given in any order. However, even for these MPCs, a node can be used only once as a dependent node. In other cases dependent degrees of freedom should not be used subsequently to impose kinematic constraints; this generally precludes the use of the first node in an MPC definition as an independent node in any

subsequent multi-point constraint, equation constraint, kinematic coupling constraint, or tie constraint definition.

Using MPCs in implicit dynamic analysis

In implicit dynamic analysis Abaqus/Standard enforces MPCs rigorously for the displacements. The velocities and accelerations are derived from the displacements with the relations defined by the dynamic integration operator (see "Implicit dynamic analysis," Section 2.4.1 of the Abaqus Theory Manual). For linear MPCs (such as PIN, TIE, and mesh refinement MPCs) and geometrically linear analysis the velocities obtained in this way satisfy the constraint exactly. However, the accelerations satisfy the constraint only approximately. If nonlinear MPCs (such as BEAM, LINK, and SLIDER) are used in geometrically nonlinear analysis, both the velocities and accelerations satisfy the constraint only approximately. In most cases the approximation is quite accurate, but in some cases high frequency oscillations may occur in the accelerations of the nodes involved in the MPC.

Using nonlinear MPCs in geometrically linear Abaqus/Standard analysis

If a nonlinear MPC is used in a geometrically linear Abaqus/Standard analysis (see "General and linear perturbation procedures," Section 6.1.2), the MPC is linearized. For example, if MPC LINK is used in a geometrically nonlinear Abaqus/Standard analysis, the distance between the two nodes of the link remains constant. If it is used in a geometrically linear Abaqus/Standard analysis, the distance between the two nodes is held constant after projection onto the direction of the line between the original positions of the nodes. The difference should be noticeable only if the magnitudes of the rotations and displacements are not small.

Defining MPCs in a user subroutine

In Abaqus/Standard you can define multi-point constraints in user subroutine **`MPC`**.

Constraints defined in user subroutine **`MPC`** can only use degrees of freedom that also exist on an element somewhere in the same model. For example, if a model contains no elements with rotational degrees of freedom, user subroutine **`MPC`** cannot use degrees of freedom 4, 5, or 6. This limitation can be overcome by adding a suitable element somewhere in the model to introduce the required degrees of freedom. This element can be added so that it does not affect the response of the model.

Constraints defined in the user subroutine are applied to the transformed degrees of freedom. A boundary nonlinearity occurs in Abaqus/Standard when MPCs are activated/deactivated in a user subroutine.

Input File Usage: *MPC, USER

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

Interaction module: **Create Connector Section**: select **MPC** as the **Connection Category** and **User-defined** as the **MPC Type**

Interaction module: **Create Constraint**: **MPC Constraint**; select **User-defined** as the **MPC Type**

Specifying the version of user subroutine `MPC`

You must specify whether the user subroutine will be coded in degree of freedom mode or in nodal mode.

Input File Usage: Use one of the following options:

*MPC, USER, MODE=DOF
*MPC, USER, MODE=NODE

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

Interaction module: **Create Connector Section**: select **MPC** as the **Connection Category** and **User-defined** as the **MPC Type**, choose **DOF-by-DOF** or **Node-by-Node**

Interaction module: **Create Constraint**: **MPC Constraint**: select **User-defined** as the **MPC Type**, choose **DOF-by-DOF** or **Node-by-Node**

Reading the data from an alternate input file

The input for an MPC definition can be contained in a separate input file.

Input File Usage: *MPC, INPUT=*file_name*

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

Abaqus/CAE Usage: Reading data from an alternate input file is not supported in Abaqus/CAE.

MPCs for mesh refinement

LINEAR — This MPC is a standard method for mesh refinement of first-order elements. It applies to all active degrees of freedom at the involved nodes including temperature, pressure, and electrical potential.

In Abaqus/Explicit it might be preferable to use a surface-based tie constraint (see "Mesh tie constraints," Section 33.3.1) for mesh refinement, particularly when one or more of the meshes to be constrained involve shell elements with thickness.

QUADRATIC(S) — This MPC is a standard method for mesh refinement of second-order elements. It applies to all active degrees of freedom at the involved nodes with the exception of temperature degrees of freedom in coupled temperature-displacement analysis and coupled thermal-electrical-structural analysis and to pressure degrees of freedom in coupled pore pressure analysis. For refinement using second-order pore pressure or coupled-temperature displacement elements, the P LINEAR or T LINEAR MPC must be used in conjunction with this MPC.

BILINEAR(S) — This MPC is a standard method for mesh refinement of first-order solid elements in three dimensions. It applies to all active degrees of freedom at the involved nodes including temperature, pressure, and electrical potential.

C BIQUAD(S) — This MPC is a standard method for mesh refinement of second-order solid elements in three dimensions. It applies to all active degrees of freedom at the

involved nodes with the exception of temperature degrees of freedom in coupled temperature-displacement analysis and coupled thermal-electrical-structural analysis and to pressure degrees of freedom in coupled pore pressure analysis. For refinement using pore pressure or coupled-temperature displacement elements in three dimensions, the P BILINEAR or T BILINEAR MPC must be used in conjunction with this MPC.

P LINEAR(S) This MPC can be used in conjunction with the QUADRATIC MPC for mesh refinement of second-order, fully coupled pore fluid flow-displacement elements. It applies to pressure degrees of freedom only. For acoustic analysis it applies the same constraint as the LINEAR MPC.

T LINEAR(S) This MPC can be used in conjunction with the QUADRATIC MPC for mesh refinement of second-order, fully coupled temperature-displacement and fully coupled thermal-electrical-structural elements. It applies to temperature degrees of freedom only. For heat transfer analysis it applies the same constraint as the LINEAR MPC.

P BILINEAR(S) This MPC can be used in conjunction with the C BIQUAD MPC for mesh refinement of pore fluid flow-displacement elements in three dimensions. It applies to pressure degrees of freedom only. For acoustic analysis it applies the same constraint as the BILINEAR MPC.

T BILINEAR(S) This MPC can be used in conjunction with the C BIQUAD MPC for mesh refinement of fully coupled temperature-displacement and fully coupled thermal-electrical-structural elements in three dimensions. It applies to temperature degrees of freedom only. For heat transfer analysis it applies the same constraint as the BILINEAR MPC.

Using mesh refinement MPCs with shell or beam elements

The Abaqus/Standard shell elements S4R5, S8R5, S9R5, and STRI65 use a penalty method to enforce transverse shear constraints on the edges of the element. The use of mesh refinement MPCs LINEAR and QUADRATIC may, therefore, lead to overconstraining or “shear locking” of the bending behavior. Graded meshes, using the triangular elements as necessary to create a transition zone, are recommended for mesh refinement with these elements.

The shear flexible beam elements in Abaqus/Standard such as B31 or B32 will also “lock” if used as stiffeners along a mesh line where the mesh refinement MPCs are used.

For shell elements in Abaqus/Explicit the rotational degrees of freedom are not constrained by the LINEAR MPC; therefore, a hinge is formed along the line defined by the constrained nodes.

Using MPC type LINEAR

MPC type LINEAR is a standard method for mesh refinement of first-order elements. However, in Abaqus/Explicit it might be preferable to use a surface-based tie constraint (see "Mesh tie constraints," Section 33.3.1) for mesh refinement, particularly when one or more of the meshes to be constrained involve shell elements with thickness.

This MPC constrains each degree of freedom at node p to be interpolated linearly from the corresponding degrees of freedom at nodes a and b (see Figure 33.2.2–1).

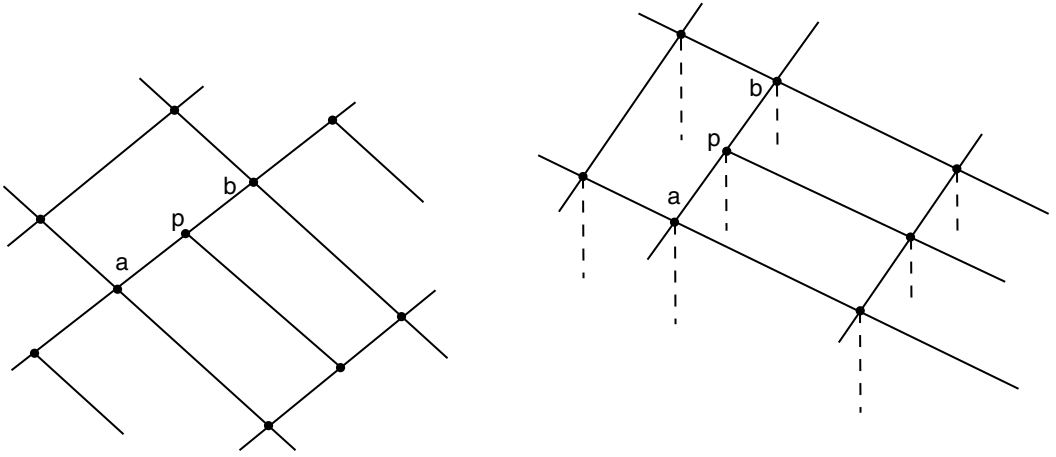


Figure 33.2.2–1 LINEAR type MPC.

Input data

Give the nodes p, a, and b as shown in Figure 33.2.2–1.

Input File Usage: *MPC
LINEAR, p, a, b

Abaqus/CAE Usage: Mesh refinement multi-point constraints are not supported in Abaqus/CAE.

Using MPC type QUADRATIC

MPC type QUADRATIC is a standard method for mesh refinement of second-order elements. This MPC type is available only in Abaqus/Standard.

This MPC constrains each degree of freedom at node ***p*** (where ***p*** is either p_1 or p_2) to be interpolated quadratically from the corresponding degrees of freedom at nodes ***a***, ***b***, and ***c*** (Figure 33.2.2–2). For coupled temperature-displacement, coupled thermal-electrical-structural, or pore pressure elements, only the displacement degrees of freedom are constrained.

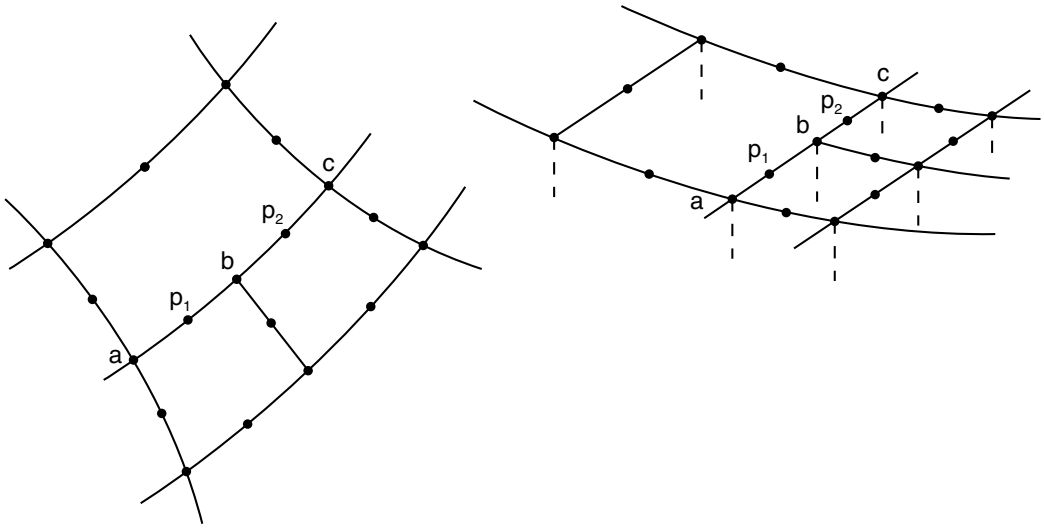


Figure 33.2.2–2 QUADRATIC type MPC.

Input data

Give the nodes ***p***, ***a***, ***b***, and ***c*** as shown in Figure 33.2.2–2, where ***p*** is either p_1 or p_2.

Input File Usage: *MPC
QUADRATIC, ***p***, ***a***, ***b***, ***c***

Abaqus/CAE Usage: Mesh refinement multi-point constraints are not supported in Abaqus/CAE.

Using MPC type BILINEAR

MPC type BILINEAR is a standard method for mesh refinement of first-order solid elements in three dimensions. This MPC type is available only in Abaqus/Standard.

This MPC constrains each degree of freedom at node p to be interpolated bilinearly from the corresponding degrees of freedom at nodes a, b, c, and d (Figure 33.2.2–3).

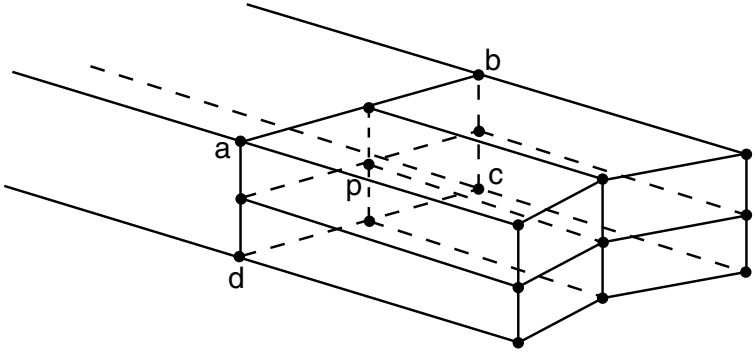


Figure 33.2.2–3 BILINEAR type MPC.

Input data

Give the nodes p, a, b, c, and d as shown in Figure 33.2.2–3.

Input File Usage: *MPC
BILINEAR, p, a, b, c, d

Abaqus/CAE Usage: Mesh refinement multi-point constraints are not supported in Abaqus/CAE.

Using MPC type C BIQUAD

MPC type C BIQUAD is a standard method for mesh refinement of second-order solid elements in three dimensions. This MPC type is available only in Abaqus/Standard.

This MPC constrains each degree of freedom at node p to be interpolated by a constrained biquadratic from the corresponding degrees of freedom at the eight nodes a, b, c, d, e, f, g, and h (Figure 33.2.2–4). For coupled temperature-displacement, coupled thermal-electrical-structural, or pore pressure elements, only the displacement degrees of freedom are constrained.

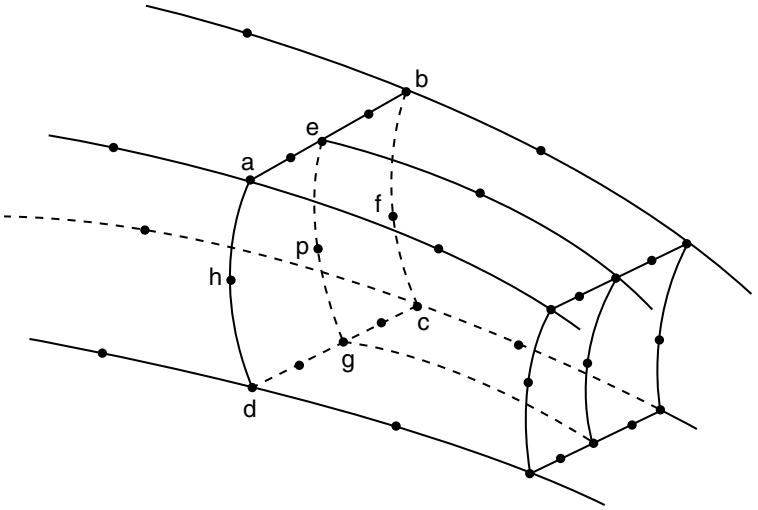


Figure 33.2.2–4 C BIQUAD type MPC.

Input data

Give the nodes p, a, b, c, d, e, f, g, and h as shown in Figure 33.2.2–4.

Input File Usage: *MPC
C BIQUAD, p, a, b, c, d, e, f, g, h

Abaqus/CAE Usage: Mesh refinement multi-point constraints are not supported in Abaqus/CAE.

Using MPC types P LINEAR and T LINEAR

The P LINEAR MPC can be used in conjunction with the QUADRATIC MPC for mesh refinement of second-order, fully coupled pore fluid flow-displacement elements.

The T LINEAR MPC can be used in conjunction with the QUADRATIC MPC for mesh refinement of second-order, fully coupled temperature-displacement and fully coupled thermal-electrical-structural elements.

These MPC types are available only in Abaqus/Standard.

These MPCs constrain the pore pressure (P LINEAR) or temperature (T LINEAR) degree of freedom at node p to be interpolated linearly from the degrees of freedom at nodes a and b (Figure 33.2.2–5).

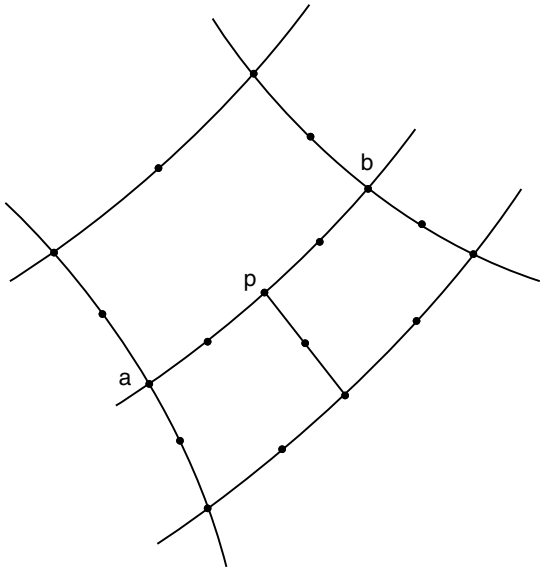


Figure 33.2.2–5 P LINEAR and T LINEAR MPCs.

Input data

Give the nodes p, a, and b as shown in Figure 33.2.2–5.

Input File Usage: Use the following option to define a P LINEAR MPC:

```
*MPC
P LINEAR, p, a, b
```

Use the following option to define a T LINEAR MPC:

```
*MPC
T LINEAR, p, a, b
```

Abaqus/CAE Usage: Mesh refinement multi-point constraints are not supported in Abaqus/CAE.

Using MPC types P BILINEAR and T BILINEAR

The P BILINEAR MPC can be used in conjunction with the C BIQUAD MPC for mesh refinement of pore fluid flow-displacement elements in three dimensions.

The T BILINEAR MPC can be used in conjunction with the C BIQUAD MPC for mesh refinement of fully coupled temperature-displacement and fully coupled thermal-electrical-structural elements in three dimensions.

These MPC types are available only in Abaqus/Standard.

These MPCs constrain the pore pressure (P LINEAR) or temperature (T LINEAR) at node *p* to be interpolated bilinearly from the pore pressure or temperature at nodes *a*, *b*, *c*, and *d* (Figure 33.2.2–6).

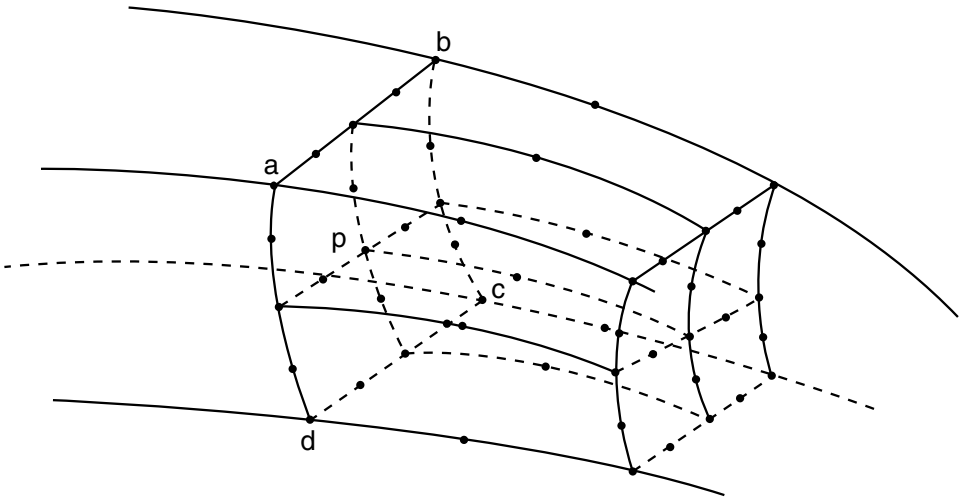


Figure 33.2.2–6 P BILINEAR and T BILINEAR MPCs.

Input data

Give the nodes *p*, *a*, *b*, *c*, and *d* as shown in Figure 33.2.2–6.

Input File Usage: Use the following option to define a P BILINEAR MPC:

```
*MPC
P BILINEAR, p, a, b, c, d
```

Use the following option to define a T BILINEAR MPC:

```
*MPC
T BILINEAR, p, a, b, c, d
```

Abaqus/CAE Usage: Mesh refinement multi-point constraints are not supported in Abaqus/CAE.

MPCs for connections and joints

BEAM	Provide a rigid beam between two nodes to constrain the displacement and rotation at the first node to the displacement and rotation at the second node, corresponding to the presence of a rigid beam between the two nodes.
CYCLSYM(S)	Constrain nodes to impose cyclic symmetry in a model.
ELBOW(S)	Constrain two nodes of ELBOW31 or ELBOW32 elements together, where the cross-sectional direction, $\mathbf{a_2}$, changes (see "Pipes and pipebends with deforming cross-sections: elbow elements," Section 28.5.1).
LINK	Provide a pinned rigid link between two nodes to keep the distance between the two nodes constant. The displacements of the first node are modified to enforce this constraint. The rotations at the nodes, if they exist, are not involved in this constraint.
PIN	Provide a pinned joint between two nodes. This MPC makes the displacements equal but leaves the rotations, if they exist, independent of each other.
REVOLUTE(S)	Provide a revolute joint.
SLIDER	Keep a node on a straight line defined by two other nodes, but allow the possibility of moving along the line and allow the line to change length.
TIE	Make all active degrees of freedom equal at two nodes.
UNIVERSAL(S)	Provide a universal joint.
V LOCAL(S)	Allow the velocity at the constrained node to be expressed in terms of velocity components at the third node defined in a local, body axis system. These local velocity components can be constrained, thus providing prescribed velocity boundary conditions in a rotating, body axis system.

See "Connectors: overview," Section 30.1.1, for element-based versions of several of these MPCs for connections and joints.

Using MPC type BEAM

MPC type BEAM provides a rigid beam between two nodes to constrain the displacement and rotation at the first node to the displacement and rotation at the second node, corresponding to the presence of a rigid beam between the two nodes.

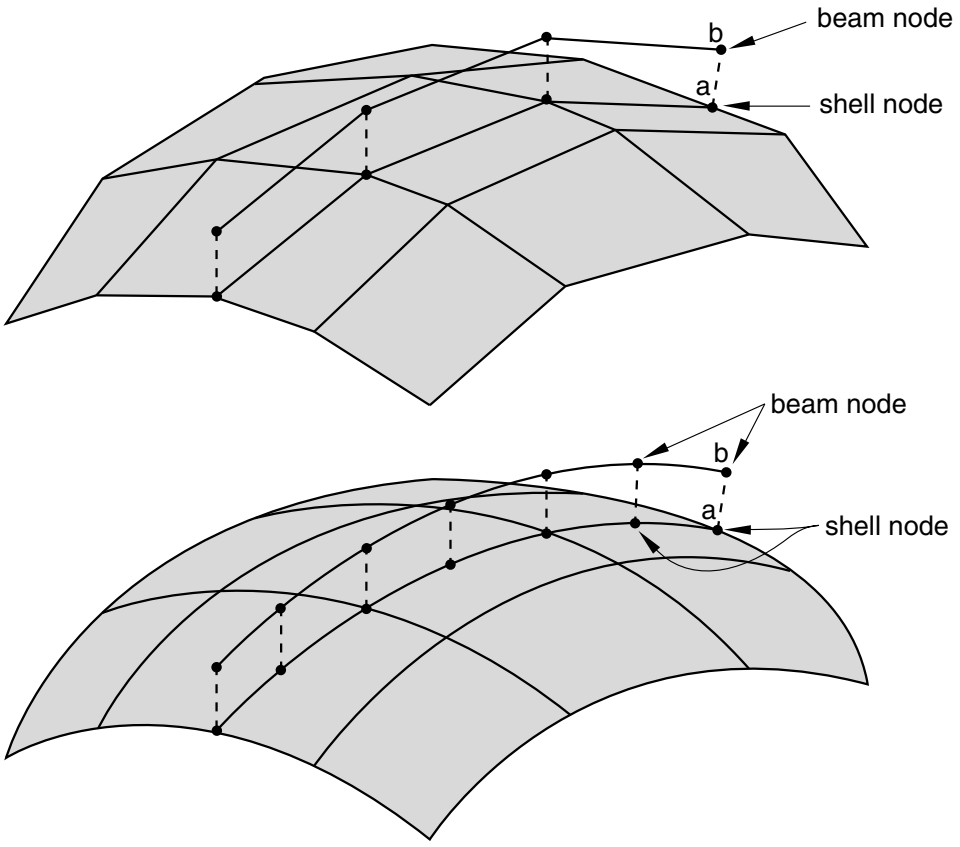


Figure 33.2.2–7 BEAM type MPC.

Input data

Give the nodes *a* and *b* as shown in Figure 33.2.2–7.

Input File Usage: ∗MPC
BEAM, *a*, *b*

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

Interaction module: **Create Connector Section**: select **MPC** as the **Connection Category** and **Beam** as the **MPC Type**

Interaction module: **Create Constraint**: **MPC Constraint**; select **Beam** as the **MPC Type**

Constraining a beam stiffener to a shell

The general method of using a beam as a stiffener on a shell is to define the beam and shell elements with separate nodes. These nodes can then be constrained to each other using BEAM type MPCs.

A more economical way, when applicable, is to use the same node for the beam node and the shell node and then define the offset of the center of the cross-section of the beam in the beam section data. Figure 33.2.2–8 shows a T-shaped stiffener attached to a shell, using the I-beam cross-section. This is done by setting l (see "Beam cross-section library," Section 28.3.9) equal to the distance between the node and the underside of the lower flange and setting the thickness of the top flange to zero. This approach can be used with all beam elements that use TRAPEZOID, I, or ARBITRARY beam sections.

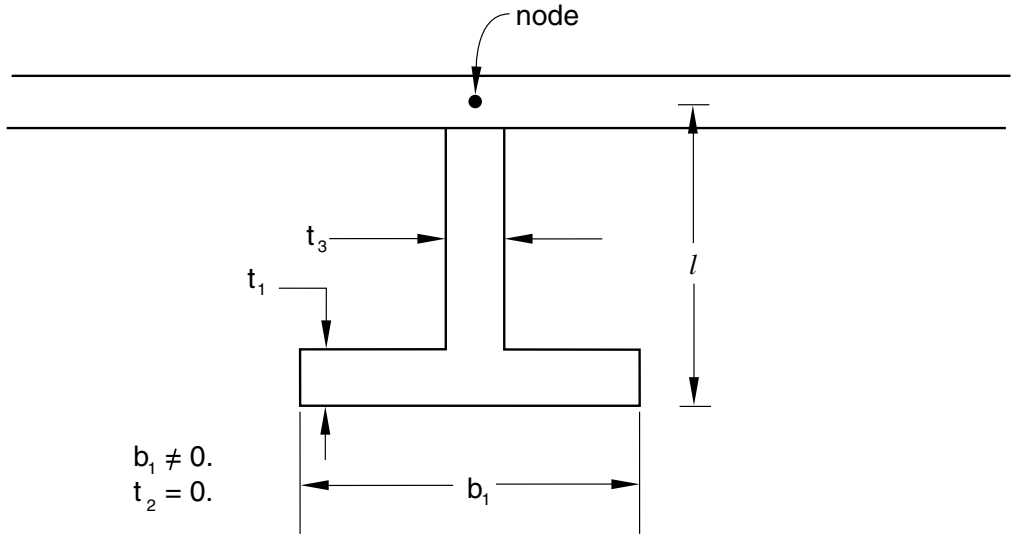


Figure 33.2.2–8 Stiffened shell.

Using MPC type CYCLSYM

MPC type CYCLSYM is used to enforce proper constraints on the radial faces bounding a segment of a cyclic symmetric structure (see Figure 33.2.2–9). This MPC type is available only in Abaqus/Standard.

MPC type CYCLSYM imposes the cyclic symmetry by equating radial, circumferential, and axial displacement components (and rotations, if active) at the two nodes (*a* and *b*). The symmetry axis can be defined by the original coordinates of two additional nodes (*c* and *d*) that do not need to be connected to any element in the structure. Scalar degrees of freedom (such as temperature) are made equal.

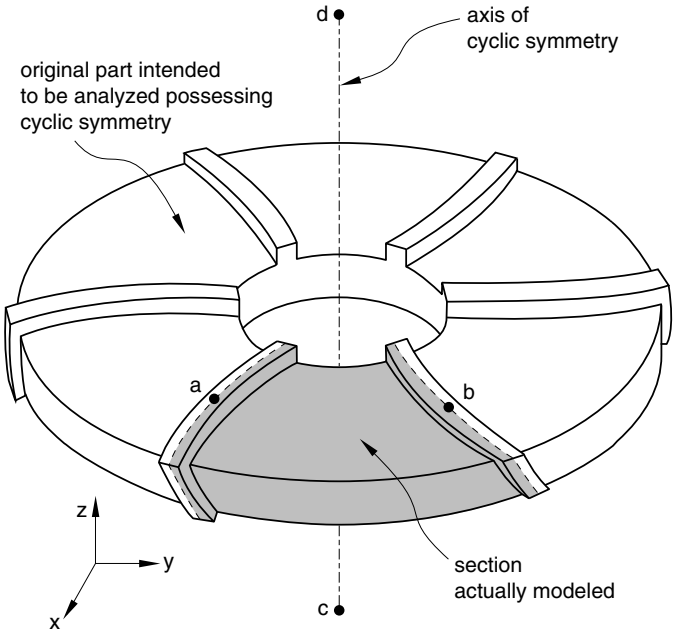


Figure 33.2.2–9 MPC type CYCLSYM.

Input data

Give the nodes *a*, *b*, and (optionally) node *c* and/or *d* that define the axis of symmetry as shown in Figure 33.2.2–9. Node set names can be used instead of the nodes *a* and *b*. If neither *c* nor *d* is given, the global *z*-axis is taken to be the axis of cyclic symmetry. If only node *c* is given, the symmetry axis passes through *c* and is parallel to the global *z*-axis. Thus, node *d* is not needed in two-dimensional cases.

Input File Usage: ∗MPC
CYCLSYM, *a*, *b*, *c*, *d*

Abaqus/CAE Usage: Cyclic symmetry multi-point constraints are not supported in Abaqus/CAE.

Using MPC type ELBOW

MPC type ELBOW constrains two nodes of ELBOW31 or ELBOW32 elements together, where the cross-sectional direction, $\mathbf{a_2}$, changes (see "Pipes and pipebends with deforming cross-sections: elbow elements," Section 28.5.1). This MPC type is available only in Abaqus/Standard.

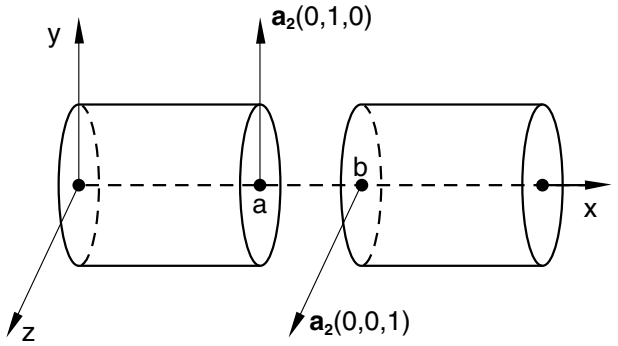


Figure 33.2.2–10 ELBOW type MPC.

Input data

Give the nodes *a* and *b* as shown in Figure 33.2.2–10.

Input File Usage: *MPC
ELBOW, *a*, *b*

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

Interaction module: **Create Connector Section**: select **MPC** as the **Connection Category** and **Elbow** as the **MPC Type**

Interaction module: **Create Constraint**: **MPC Constraint**; select **Elbow** as the **MPC Type**

Using MPC type LINK

MPC type LINK provides a pinned rigid link between two nodes to keep the distance between the nodes constant, as shown in Figure 33.2.2–11. The displacements of the first node are modified to enforce this constraint. The rotations at the nodes, if they exist, are not involved in this constraint.

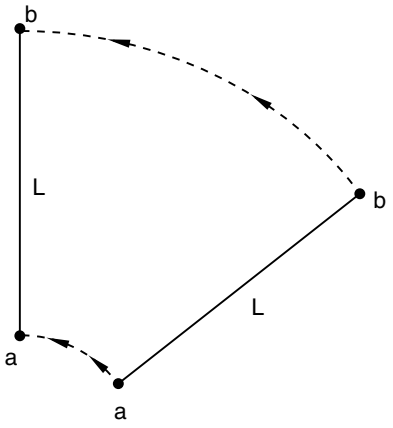


Figure 33.2.2–11 MPC type LINK.

Input data

Give the nodes *a* and *b* as shown in Figure 33.2.2–11.

Input File Usage: *MPC
LINK, *a*, *b*

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

Interaction module: **Create Connector Section**: select **MPC** as the **Connection Category** and **Link** as the **MPC Type**

Interaction module: **Create Constraint**: **MPC Constraint**; select **Link** as the **MPC Type**

Using MPC type PIN

MPC type PIN provides a pinned joint between two nodes. This MPC makes the global displacements equal but leaves the rotations, if they exist, independent of each other, as shown in Figure 33.2.2–12.

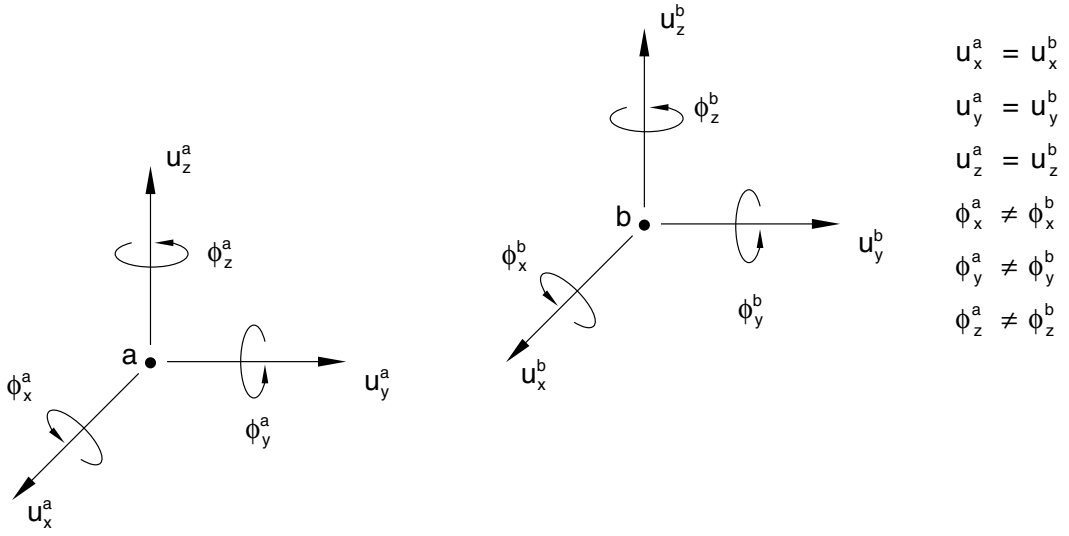


Figure 33.2.2–12 MPC type PIN.

Input data

Give the nodes *a* and *b* as shown in Figure 33.2.2–12.

Input File Usage: *MPC
PIN, *a*, *b*

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

Interaction module: **Create Connector Section**: select **MPC** as the **Connection Category** and **Pin** as the **MPC Type**

Interaction module: **Create Constraint**: **MPC Constraint**; select **Pin** as the **MPC Type**

Using MPC type REVOLUTE

This MPC type is available only in Abaqus/Standard.

A revolute joint is a joint in which relative rotation is allowed between two nodes about an axis that rotates during the motion (see Figure 33.2.2–13). The axis of the joint is defined in the initial configuration as the line from node *b* to node *c*. If these nodes are coincident, the axis is assumed to be the global *z*-axis. The rotation of the joint axis is that of node *b*.

The relative rotation in the joint is a single variable and is stored as degree of freedom 6 at node *c*. This degree of freedom can be used with other members in the model, but caution should be used because of the nonstandard use of degree of freedom 6. For example, a SPRING1 element (a spring to ground) might be attached to this degree of freedom. Since the degree of freedom measures a *relative* rotation, this spring would then be a torsional spring between nodes *a* and *b*.

The displacements at node *a* are not constrained by the REVOLUTE MPC to be the same as the displacements at node *b*. Thus, the joint definition must usually be completed either by using a PIN type MPC between nodes *a* and *b* or by using suitable stiffness members between these two nodes.

An example of a revolute joint and application of the REVOLUTE MPC is provided in "Revolute MPC verification: rotation of a crank," Section 1.3.8 of the Abaqus Benchmarks Manual. See "Revolute joint," Section 6.6.3 of the Abaqus Theory Manual, for more details on revolute joints.

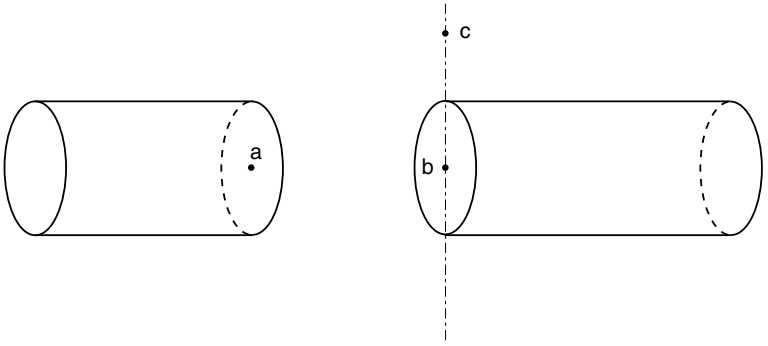


Figure 33.2.2–13 Revolute joint.

Input data

Give the nodes *a*, *b*, and *c* as shown in Figure 33.2.2–13. Degree of freedom 6 at node *c* defines the *relative* rotation between nodes *a* and *b*; therefore, this degree of freedom does not obey the standard convention for degrees of freedom in Abaqus.

Input File Usage: *MPC
REVOLUTE, *a*, *b*, *c*

Abaqus/CAE Usage: Revolute joint multi-point constraints are not supported in Abaqus/CAE.

Using MPC type SLIDER

MPC type SLIDER keeps a node on a straight line defined by two other nodes but allows the possibility of moving along the line and allows the line to change length.

When transitioning from multiple layers of solid elements to shells, it is often desirable to constrain the nodes on the free edge of the solid elements to remain in a straight line. (This constraint is consistent with shell theory.) The SLIDER MPC can perform this function without restraining the "thinning" behavior of the solid layers. The SS LINEAR MPC is then used to attach the shell element to this edge.

In Abaqus/Standard when a SLIDER MPC is used with one of the shell-solid MPCs—SS LINEAR, SS BILINEAR, or SSF BILINEAR—it must be given following the shell-solid MPCs.

Input data

For each node p shown in Figure 33.2.2–14 and Figure 33.2.2–15, give the nodes p, a, and b for each line of nodes that should remain straight. For each node q shown in Figure 33.2.2–14, give the nodes q, c, and d, and so on for each line of nodes that should remain straight.

Input File Usage: *MPC
SLIDER, p, a, b
SLIDER, q, c, d

Abaqus/CAE Usage: Slider multi-point constraints are not supported in Abaqus/CAE.

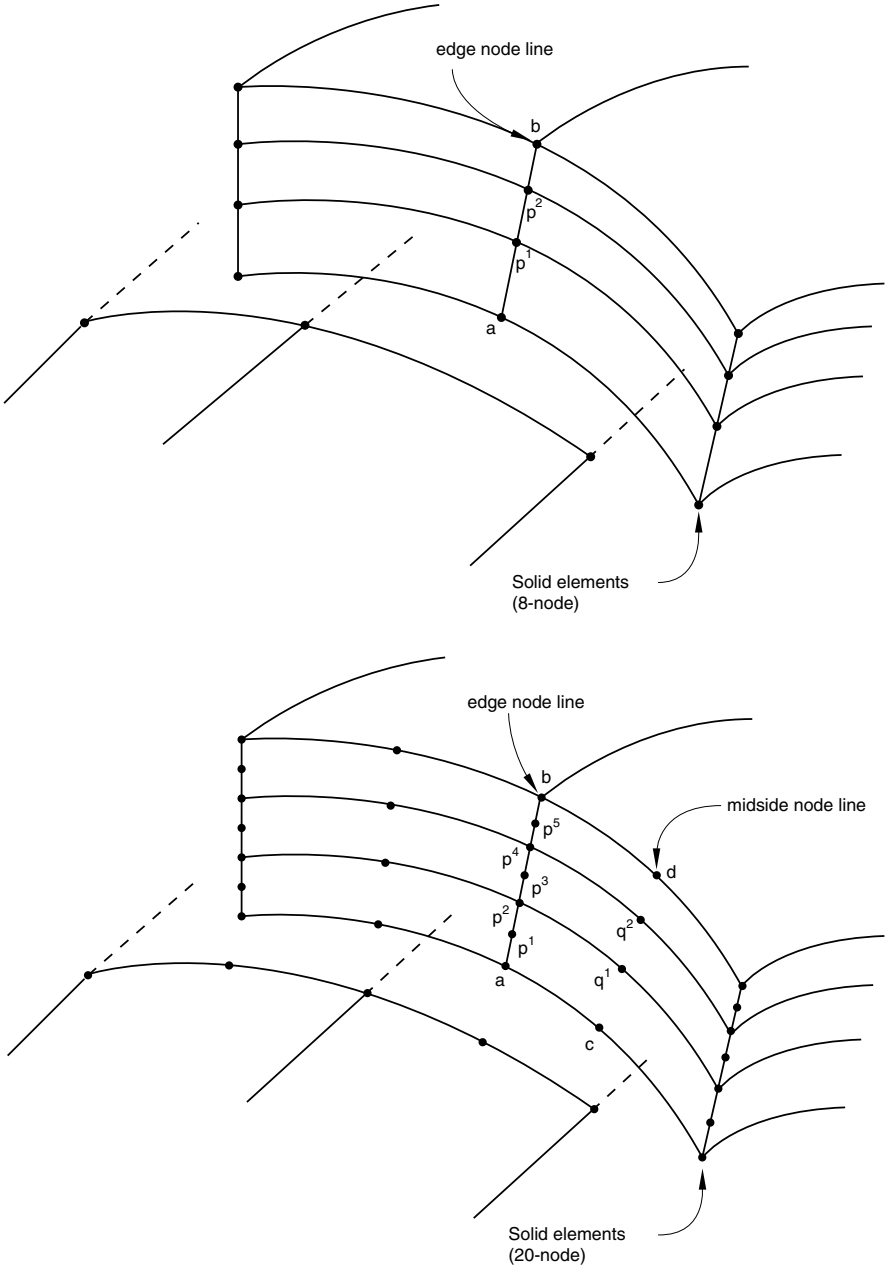


Figure 33.2.2–14 SLIDER type MPC used at a shell-solid intersection.

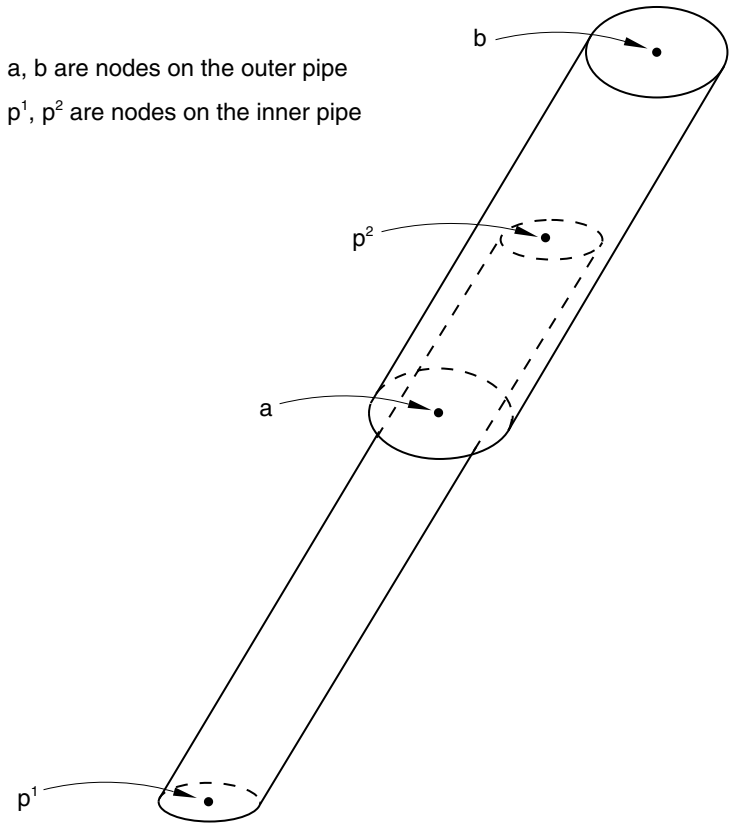


Figure 33.2.2–15 SLIDER type MPC used to model a telescoping beam.

Using MPC type TIE

MPC type TIE makes the global displacements and rotations as well as all other active degrees of freedom equal at two nodes. If there are different degrees of freedom active at the two nodes, only those in common will be constrained.

MPC type TIE is usually used to join two parts of a mesh when corresponding nodes on the two parts are to be fully connected ("zipping up" a mesh). For example, when a mesh is generated on a cylindrical body, the solution at the nodes at 0° and those at 360° must be the same. This can be done either by renumbering the nodes on one of the mesh extremes or by using this MPC for each pair of corresponding nodes, as shown in Figure 33.2.2–16.

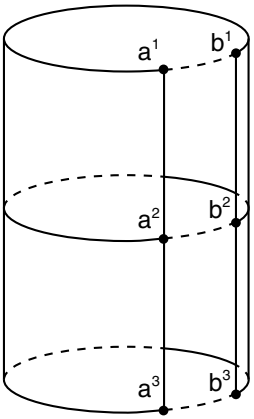


Figure 33.2.2–16 Example of use of TIE MPC.

Input data

Give the nodes *a* and *b* as shown in Figure 33.2.2–16.

Input File Usage: *MPC
TIE, *a*, *b*

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

Interaction module: **Create Connector Section**: select **MPC** as the **Connection Category** and **Tie** as the **MPC Type**

Interaction module: **Create Constraint**: **MPC Constraint**; select **Tie** as the **MPC Type**

Using MPC type UNIVERSAL

This MPC type is available only in Abaqus/Standard.

A universal joint is a joint in which relative rotation is allowed between two nodes, about two axes that are connected rigidly, and each of which rotates with the rotation of one end of the joint (see Figure 33.2.2–17). Such a joint might be used to couple two shafts that have an angular misalignment. The first axis of the joint, which is attached to node *b*, is defined in the initial configuration as the line from node *b* to node *c*. If these nodes are coincident, the axis is assumed to be the global *z*-axis. The second axis of the joint is at right angles to the first axis and is in the plane defined by the first axis and node *d*.

The relative rotations in the joint are stored as degree of freedom 6 at the nodes *c* and *d*. These degrees of freedom can be used with other members in the model, but caution should be used because of the nonstandard use of degree of freedom 6. For example, a SPRING1 element (a spring to ground) might be attached to one of these degrees of freedom. Since the degree of freedom measures a *relative* rotation, this spring would then be a torsional spring, restraining that component of relative rotation.

The displacements at node *a* are not constrained by the UNIVERSAL MPC to be the same as the displacements at node *b*. Thus, the joint definition must usually be completed either by using a PIN type MPC between nodes *a* and *b* or by using suitable stiffness members between these two nodes.

See "Universal joint," Section 6.6.4 of the Abaqus Theory Manual, for more details on universal joints.

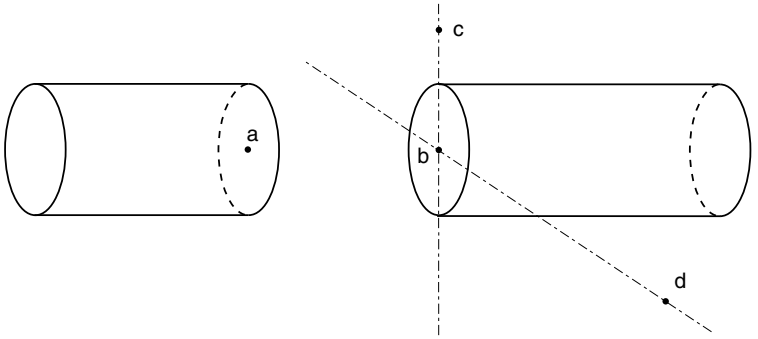


Figure 33.2.2–17 Universal joint.

Input data

Give the nodes *a*, *b*, *c*, and *d* as shown in Figure 33.2.2–17. Degrees of freedom 6 at nodes *c* and *d* define the *relative* rotation in the joint; therefore, these degrees of freedom do not obey the standard convention for degrees of freedom in Abaqus.

Input File Usage: *MPC
UNIVERSAL, *a*, *b*, *c*, *d*

Abaqus/CAE Usage: Universal joint multi-point constraints are not supported in Abaqus/CAE.

Using MPC type V LOCAL

This MPC type is available only in Abaqus/Standard.

As shown in Figure 33.2.2–18, MPC type V LOCAL constrains the velocity components associated with degrees of freedom 1, 2, and 3 at a first node (*a*) to be equal to the velocity components at a third node (*c*) along local, rotating directions. These local directions rotate according to the rotation at a second node (*b*). In the initial configuration the first local direction is from the second to the third node of the MPC (from *b* to *c*, as indicated by the arrows in Figure 33.2.2–18), or it is the global *z*-axis if these nodes coincide. The other local directions are then defined by the standard Abaqus convention for such directions (see "Conventions," Section 1.2.2). In Figure 33.2.2–18 this MPC is applied to nodes *d*, *e*, and *f* in the same manner.

MPC type V LOCAL can be useful for defining a complex motion within a model. For example, the MPC can be used to model the steering of an automobile in a dynamic analysis for which the resulting inertial effects are of interest. See "Local velocity constraint," Section 6.6.5 of the Abaqus Theory Manual, for more details on the local velocity constraint.

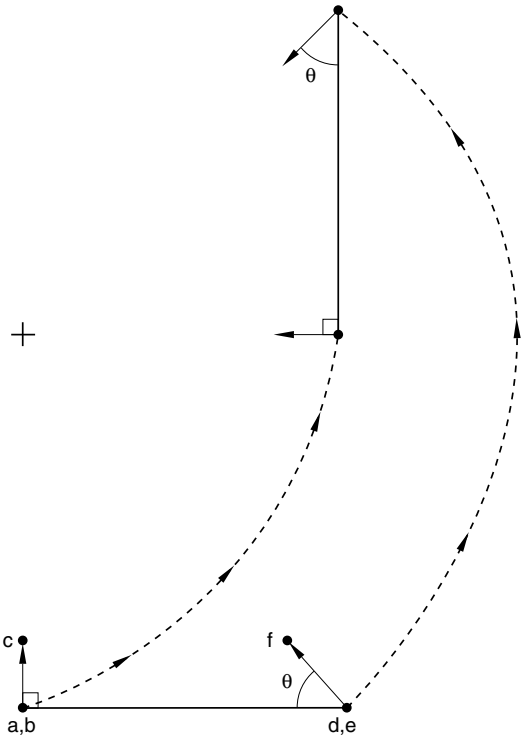


Figure 33.2.2–18 Local velocity constraint.

Input data

Give the node whose velocity components are constrained (node *a* or *d* in Figure 33.2.2–18), the node whose rotation defines the rotation of the local directions (node *b* or *e* in Figure 33.2.2–18), and the node whose velocity components are in these local directions (node *c* or *f* in Figure 33.2.2–18). Nodes *a* and *b* (or *d* and *e*) can be the same.

Input File Usage: *MPC
V LOCAL, *a*, *b*, *c*
V LOCAL, *d*, *e*, *f*

Abaqus/CAE Usage: Local velocity component multi-point constraints are not supported in Abaqus/CAE.

MPCs for transitions

SS LINEAR	Constrain a shell node to a solid node line for linear elements (S4, S4R, S4R5, C3D8, C3D8R, SAX1, CAX4, etc.).
SS BILINEAR(S)	Constrain a shell node to a solid node line for edge lines on quadratic elements (S8R, S8R5, C3D20, C3D20R, SAX2, CAX8, etc.).
SSF BILINEAR(S)	Constrain a midside node of a quadratic shell element (S8R, S8R5) to midface lines on 20-node bricks (C3D20, C3D20R, etc.).

Modeling a shell-to-solid element transition

The SLIDER, SS LINEAR, SS BILINEAR, and SSF BILINEAR MPCs allow for a transition from shell element modeling to solid element modeling on a shell surface. This modeling technique can be used to obtain solutions at shell-solid intersections or other discontinuities, where the local modeling should use full three-dimensional theory but the other parts of the structure can be modeled as shells. The shell-to-solid submodeling capability ("Submodeling: overview," Section 10.2.1) and the surface-based shell-to-solid coupling constraint ("Shell-to-solid coupling," Section 33.3.3) can also be used to obtain more accurate solutions in such cases, with considerably less modeling effort.

In Abaqus/Standard the MPC usage assumes that the interface between the shell and solid elements is a surface containing the normals to the shell along the line of intersection of the meshes, so that the lines of nodes on the solid mesh side of the interface in the normal direction to the surface are straight lines. (Line a, p^1, p^2, …, b in Figure 33.2.2–14 and lines p^1, p^2, …, p^n in Figure 33.2.2–19 to Figure 33.2.2–20 should be straight lines.) It also assumes that the nodes of the solid elements are spaced uniformly on the interface surface as indicated in Figure 33.2.2–14 and Figure 33.2.2–19 to Figure 33.2.2–20. For each shell node on the edge use MPC type SS LINEAR, SS BILINEAR, or SSF BILINEAR, as appropriate, to constrain the shell node to the corresponding line or face of solid element nodes through the thickness. Then, use a SLIDER MPC to constrain each interior node on the line through the thickness to remain on the straight line defined by the bottom and top nodes of that line. For an example, see "*MPC," Section 5.1.17 of the Abaqus Verification Manual.

The SS BILINEAR and SSF BILINEAR MPCs are not intended for use with the variable node solid elements (C3D27, C3D27H, C3D27R, and C3D27RH).

In Abaqus/Standard MPCs SS LINEAR, SS BILINEAR, and SSF BILINEAR eliminate all displacement components and two of the rotation components at the shell node, and the SLIDER MPC eliminates two displacement components at each interior solid element node in the interface. Therefore, any boundary conditions needed at the interface (such as those required when the shell/solid interface intersects a symmetry plane) should be applied only to the top and bottom nodes on the solid element side of the interface.

Using MPC type SS LINEAR

MPC type SS LINEAR constrains a shell corner node to a line of edge nodes on solid elements for linear elements (S4, S4R, or S4R5; C3D8, C3D8R; SAX1; CAX4; etc.).

The constrained nodes need not lie exactly on these lines, but it is suggested that they be in close proximity to the lines for meaningful results.

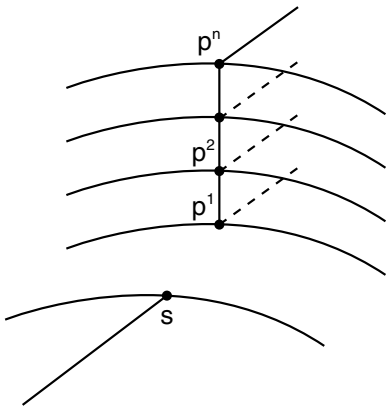


Figure 33.2.2–19 SS LINEAR type MPC. 4-node shells to 8-node bricks.

Input data

Give the shell node, S, then the list of nodes along the corresponding line through the thickness in the solid element mesh. In Abaqus/Explicit only two solid nodes can be given. Referring to Figure 33.2.2–19, in Abaqus/Standard give S, p^1, p^2, …, p^n, and in Abaqus/Explicit give S, p^1, p^n, where $n \geq 2$. The shell node number must be different from the solid mesh node numbers.

Input File Usage: In Abaqus/Standard use the following option:

```
*MPC
 SS LINEAR, S, p^1, p^2, …, p^n
```

In Abaqus/Explicit use the following option:

```
*MPC
 SS LINEAR, S, p^1, p^n
```

Abaqus/CAE Usage: Multi-point constraints for transitions are not supported in Abaqus/CAE.

Using MPC type SS BILINEAR

MPC type SS BILINEAR constrains a corner node of a quadratic shell element (S8R, S8R5) to a line of edge nodes on 20-node bricks. This MPC type is available only in Abaqus/Standard.

The constrained node need not lie exactly on the line, but it is suggested that it be in close proximity to the line for meaningful results.

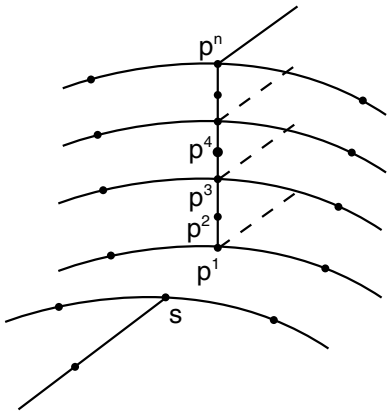


Figure 33.2.2–20 SS BILINEAR type MPC. Corner of 8-node shell to edge of 20-node bricks.

Input data

Give the shell node, S, then the list of nodes along the corresponding line through the thickness in the solid element mesh. Referring to Figure 33.2.2–20, give S, p^1, p^2,..., p^n. The shell node number must be different from the solid mesh node numbers.

Input File Usage: ∗MPC
SS BILINEAR, S, p^1, p^2, ..., p^n

Abaqus/CAE Usage: Multi-point constraints for transitions are not supported in Abaqus/CAE.

Using MPC type SSF BILINEAR

MPC type SSF BILINEAR constrains a midside node on a quadratic shell element (S8R, S8R5) to a line of midface nodes on solid 20-node bricks. This MPC type is available only in Abaqus/Standard.

The constrained node need not lie exactly on the line, but it is suggested that it be in close proximity to the line for meaningful results.

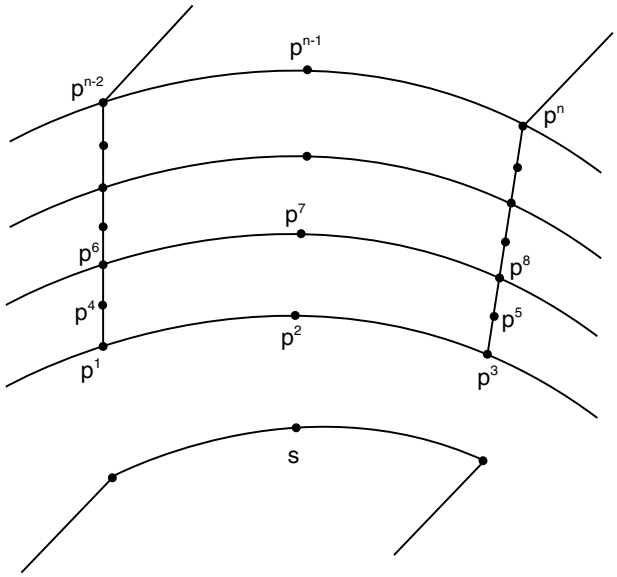


Figure 33.2.2–21 SSF BILINEAR type MPC. Midside of 8-node shell to surface of 20-node bricks.

Input data

Give the shell node, S, then the list of nodes on the solid face, in the order p^1, p^2,…, p^n as shown in Figure 33.2.2–21.

Input File Usage: *MPC
SSF BILINEAR, S, p^1, p^2, …, p^n

Abaqus/CAE Usage: Multi-point constraints for transitions are not supported in Abaqus/CAE.

33.2.3 KINEMATIC COUPLING CONSTRAINTS

Product: Abaqus/Standard

References

- "Kinematic constraints: overview," Section 33.1.1
- *KINEMATIC COUPLING

Overview

Kinematic coupling constraints:

- limit the motion of a group of nodes to the rigid body motion defined by a reference node;
- can be applied only to specific user-specified degrees of freedom at the constrained nodes;
- can be specified with respect to local coordinate systems at the constrained nodes; and
- can be used in geometrically linear or nonlinear analysis.

The preferred method of providing a kinematic constraint of this type is described in "Coupling constraints," Section 33.3.2.

Typical applications

The kinematic coupling constraints are useful in cases where a large number of nodes (the "coupling" nodes) are constrained to the rigid body motion of a single node and the degrees of freedom that participate in the constraint are selected individually in a local coordinate system. In many such cases MPCs either are not available or would have to be prescribed individually for each constrained node. A typical example is shown in Figure 33.2.3–1, where a kinematic coupling constraint is used to prescribe a twisting motion to a model without constraining radial motions. In other applications the kinematic coupling constraint can be used to provide coupling between continuum and structural elements.

Defining the constraint

A kinematic coupling constraint requires the specification of a reference node, coupling nodes, and the constrained degrees of freedom at these nodes. The reference node has both translational and rotational degrees of freedom.

Kinematic constraints are imposed by eliminating degrees of freedom at the coupling nodes. Once any combination of displacement degrees of freedom at a coupling node is constrained, additional displacement constraints—such as MPCs, boundary conditions, or other kinematic coupling definitions—cannot be applied to any coupling node involved in a kinematic coupling constraint. The same limitation applies for rotational degrees of freedom.

Input File Usage: To constrain all available degrees of freedom:

*KINEMATIC COUPLING, REF NODE=*node*
coupling node number or node set

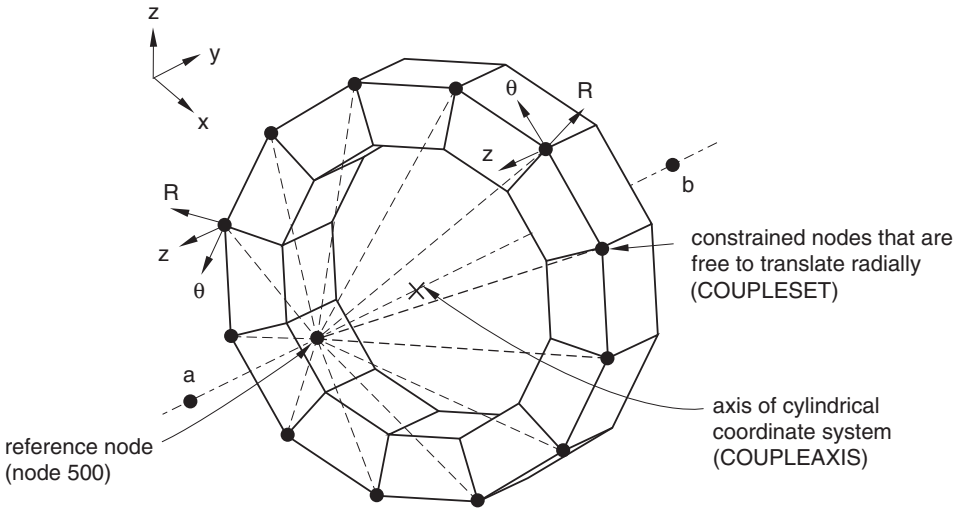


Figure 33.2.3–1 A kinematic coupling constraint used to transmit rotation to a structure while permitting radial motion.

To constrain a single degree of freedom:

∗KINEMATIC COUPLING, REF NODE=*node*
coupling node number or node set, dof

To constrain a range of degrees of freedom:

∗KINEMATIC COUPLING, REF NODE=*node*
coupling node number or node set, first dof, last dof

To specify non-contiguous lists of constrained degrees of freedom, repeat the node numbers or node sets on subsequent data lines. For example, the following input is used to constrain degrees of freedom 1, 2, 3, and 6 at node 10 to the motion of reference node 5:

```
*KINEMATIC COUPLING, REF NODE=5
10, 1, 3
10, 6
```

Translational degrees of freedom

Translational degrees of freedom are constrained by eliminating the specified degrees of freedom at the coupling nodes. When all translational degrees of freedom are specified, the coupling nodes follow the rigid body motion of the reference node.

Rotational degrees of freedom

All combinations of selected rotational degrees of freedom result in rotational behavior that is identical to existing MPC types. Specifically:

- Selection of three rotational degrees of freedom along with three displacement degrees of freedom is equivalent to MPC type BEAM.
- Selection of two rotational degrees of freedom is equivalent to MPC type REVOLUTE.
- Selection of one rotational degree of freedom is equivalent to MPC type UNIVERSAL.

Internal nodes are created by the kinematic coupling to enforce the constraints that are equivalent to MPC types REVOLUTE and UNIVERSAL. These nodes have the same degrees of freedom as the additional nodes used in these MPC types and are included in the residual check for nonlinear analysis.

Specifying a local coordinate system

The constrained degrees of freedom at the coupling nodes can be specified in a local coordinate system instead of the (default) global coordinate system (see "Orientations," Section 2.2.5). Figure 33.2.3–1 illustrates the use of a local coordinate system definition with a kinematic coupling constraint to constrain all but the radial translation of a group of nodes to a reference node. In this example a local cylindrical coordinate system is defined that has its axis coincident with the structure's axis. The coupling node constraints are then specified in this local coordinate system. In this example the constrained nodes are attached to continuum elements; thus, only translational degrees of freedom need to be specified.

Input File Usage: *KINEMATIC COUPLING, REF NODE=*node*, ORIENTATION=*name*

For example, the following input is used to specify the kinematic coupling constraint shown in Figure 33.2.3–1:

```
*ORIENTATION, SYSTEM=CYLINDRICAL, NAME=COUPLEAXIS
0.0, -1.0, 0.0, 0.0, 1.0, 0.0
*KINEMATIC COUPLING, REF NODE=500,
ORIENTATION=COUPLEAXIS
COUPLESET, 2, 3
```

Constraint directions and finite rotations

In geometrically nonlinear analysis steps, the coordinate system in which the constrained degrees of freedom are specified will rotate with the reference node regardless of whether the constrained degrees of freedom are specified in the global coordinate system or in a local system. Thus, the constraint shown in Figure 33.2.3–1 will enable free radial motion throughout arbitrary rotations of the structure. Radial motion in this case is defined as motion normal to the structure's axis (defined in the undeformed configuration by points a and b in the figure), with this axis rotating with the reference node. Therefore, the free radial expansion shown in Figure 33.2.3–1 will not refer to an axis parallel to the global y-axis for general rotations of the reference node but will refer to an axis that rotates with the structure. Rotation of the constraint directions is not affected by the selection of the constrained degrees of freedom.

33.3 Surface-based constraints

- “Mesh tie constraints,” Section 33.3.1
- “Coupling constraints,” Section 33.3.2
- “Shell-to-solid coupling,” Section 33.3.3
- “Mesh-independent fasteners,” Section 33.3.4

33.3.1 MESH TIE CONSTRAINTS

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Surfaces: overview," Section 2.3.1
- *TIE
- "Defining tie constraints," Section 15.15.1 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Using contact and constraint detection," Section 15.16 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

A surface-based tie constraint:

- ties two surfaces together for the duration of a simulation;
- can be used only with surface-based constraint definitions;
- can be used in mechanical, coupled temperature-displacement, coupled thermal-electrical-structural, acoustic pressure, coupled acoustic pressure-displacement, coupled pore pressure–displacement, coupled thermal-electrical, or heat transfer simulations;
- can also be used to create a constraint on a surface so that it follows the motion of a three-dimensional beam;
- is useful for mesh refinement purposes, especially for three-dimensional problems;
- allows for rapid transitions in mesh density within the model;
- constrains each of the nodes on the slave surface to have the same motion and the same value of temperature, pore pressure, acoustic pressure, or electrical potential as the point on the master surface to which it is closest;
- will take the initial thickness and offset of shell elements underlying the surface into account by default; and
- eliminates the degrees of freedom of the slave surface nodes that are constrained, where possible.

Defining a tie constraint for a pair of surfaces

A surface-based tie constraint can be used to make the translational and rotational motion as well as all other active degrees of freedom equal for a pair of surfaces. By default, as discussed below, nodes are tied only where the surfaces are close to one another. One surface in the constraint is designated to be the slave surface; the other surface is the master surface. A name must be assigned to this constraint and may be used in postprocessing with Abaqus/CAE.

Input File Usage: *TIE, NAME=*name*
slave_surface_name, *master_surface_name*

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Tie**

Defining the surfaces to be constrained

Either element-based or node-based surfaces can be used as the slave surface. Any surface type (element-based, node-based, or analytical) can be used as the master surface. You may need to take some surface restrictions into consideration depending on which tie formulation is used and whether the analysis is conducted in Abaqus/Standard or Abaqus/Explicit. Two tie formulations are available: the surface-to-surface formulation, which is used by default in Abaqus/Standard, and the more traditional node-to-surface formulation, which is used by default in Abaqus/Explicit; these formulations are discussed in more detail later in this section. Table 33.3.1–1 and Table 33.3.1–2 provide comparisons of surface restrictions for the different formulations and analysis codes.

Table 33.3.1–1 Comparison of characteristics for surface-based tie formulations.

Tie formulation	Optimized stress accuracy	Node-based surfaces allowed	Mixture of rigid and deformable subregions allowed	Treatment of nodes/facets shared between master and slave surfaces
Surface-to-surface (Abaqus/Standard or Abaqus/Explicit)	Yes	Reverts to node-to-surface formulation	No	Eliminated from slave
Node-to-surface in Abaqus/Standard	No	Yes	No	Eliminated from slave
Node-to-surface in Abaqus/Explicit	No	Yes	Yes	Eliminated from master

The surface-to-surface formulation generally avoids stress noise at tied interfaces. As indicated in Table 33.3.1–1 and Table 33.3.1–2, only a few surface restrictions apply to the surface-to-surface formulation: this formulation reverts to the node-to-surface formulation if a node-based or edge-based surface is used. The surface-to-surface formulation does not allow for a mixture of rigid and deformable portions of a surface, and the master surface must not contain T-intersections. Any nodes shared between the slave and master surfaces will not be tied with the surface-to-surface formulation. The same comments apply to both Abaqus/Standard and Abaqus/Explicit in these tables for the surface-to-surface formulation.

With the more traditional node-to-surface formulation additional surface restrictions apply in Abaqus/Standard but fewer restrictions apply in Abaqus/Explicit in comparison to the surface-to-surface

Table 33.3.1–2 Comparison of element-based surface characteristics allowed for surface-based tie formulations.

Tie formulation	Surface Characteristics (Yes=allowed, No=not allowed)			
	Double-sided	Discontinuous	T-intersection	Edge-based
Surface-to-surface (Abaqus/Standard or Abaqus/Explicit)	Master: Yes Slave: Yes	Master: Yes Slave: Yes	Master: No Slave: Yes	Reverts to node-to-surface formulation if either surface is edge-based
Node-to-surface in Abaqus/Standard	Master: Yes Slave: Yes	Master: Yes Slave: Yes	Master: No Slave: Yes	Master: Yes Slave: Yes
Node-to-surface in Abaqus/Explicit	Master: Yes Slave: Yes	Master: Yes Slave: Yes	Master: Yes Slave: Yes	Master: Yes Slave: Yes

formulation. Relatively stringent restrictions on master surface connectivity for the node-to-surface tie formulation in Abaqus/Standard are indicated in Table 33.3.1–2: the master surface must be simply connected and must not contain complex intersections such as T-intersections (see "Defining contact pairs in Abaqus/Standard," Section 34.3.1, for examples of surfaces with various connectivity characteristics).

Differences with the node-to-surface formulation in Abaqus/Explicit are apparent in Table 33.3.1–1: partially rigid surfaces can be used and the treatment of shared portions of slave and master surfaces is unique to this case. Nodes and faces that are shared between the master and slave surfaces are eliminated automatically from the master surface in this case if the paired surfaces are either both element-based or both node-based, enabling the possibility of tying multiple slave surfaces (defined over various regions of the model) to a common master surface defined over the entire model. This is a convenient way to define tie constraints in large models, as it eliminates the need for defining specialized master surfaces for each surface pairing; however, you must still take care that slave surfaces do not include portions of the opposing surface to which they should be tied (for example, no tie constraints will be generated if the master and slave surfaces are identical). In the node-to-surface formulation in Abaqus/Explicit all facets attached to nodes that are common between slave and master surfaces are excluded from being tied to slave nodes. Sometimes when meshes are transitioned from one type of element to another type or from one element size to another element size, common nodes may exist at the interface of the two regions. Typically, a tie constraint is defined at the interface of the two zones to stitch the two meshes together. In a situation like this common nodes may get tied to a neighboring facet on the interface and may cause undesirable mesh distortion due to the tie adjustment. One possible way to avoid the undesirable mesh distortion is to specify a very small position tolerance for the tie pair. Another situation that may arise when common nodes occur between the slave and master surfaces at the interface of mesh transition zones is that slave nodes in the vicinity of the common node may not get tied. This happens due to the exclusion of master facets attached to the common nodes. Therefore, care must be taken to ensure that

elements in different mesh zones do not share common nodes at the interface. For all such common nodes, duplicate nodes occupying the same physical location should be defined.

Input File Usage: Use the *SURFACE option to define the slave and master surfaces used in the constraint (see "Surfaces: overview," Section 2.3.1):

*SURFACE, NAME=*slave_surface_name*
*SURFACE, NAME=*master_surface_name*

Abaqus/CAE Usage: In Abaqus/CAE you can select one or more faces directly in the viewport when you are prompted to select a surface. In addition, you can define surfaces as collections of faces and edges using the Surface toolset.

Specifying the subset of slave nodes to be constrained

By default, Abaqus uses a position tolerance criterion to determine the constrained nodes based on the distance between the slave nodes and the master surface. Alternatively, you can specify a node set containing the slave nodes to be constrained regardless of their distance to the master surface.

Using the position tolerance criterion

The default position tolerance criterion ensures that nodes are tied only where the slave and master surfaces are close to one another in the initial configuration. For example, consider the case shown in Figure 33.3.1–1. Surfaces **`Comp1_surf`** and **`Comp2_surf`** are defined to cover all exposed faces of Component 1 and Component 2, respectively. These two surfaces can be used as the slave and master surfaces in a tie constraint to tie the two components in the desired region, because only the nodes at the initial interface between the two surfaces are tied.

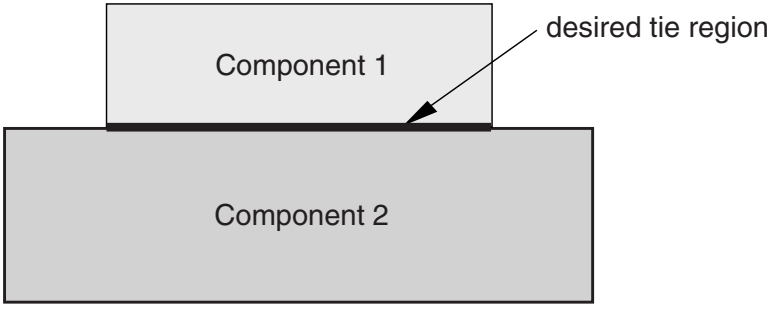


Figure 33.3.1–1 Example of two components to be tied together.

The default value of the position tolerance, d_{tol}, typically results in desired tie constraints with little effort. Details regarding the calculation of distances between surfaces and default values of the position tolerances are provided below. You can modify the position tolerance if desired.

Input File Usage: Use the following option to use the default position tolerance:

∗TIE

Use the following option to specify a position tolerance:

∗TIE, POSITION TOLERANCE=*distance*

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Tie**: **Position Tolerance**: **Specify distance**

Calculating the distance between surfaces

The following factors influence the calculation of the distance between surfaces for a particular slave node:

- Shell thickness. By default, calculations of distances between surfaces account for shell thickness and offset effects for element-based slave or master surfaces: the distance is measured from the actual top or bottom side of the surface, whichever is closer to the other surface. Alternatively, you can specify that surface thicknesses and offsets should be ignored, which also has implications for nodal position adjustments for resolving initial gaps (discussed later).

  Input File Usage: Use the following option to ignore surface thicknesses and offsets in the distance calculations:

  ∗TIE, NO THICKNESS

  Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Tie**: **Exclude shell element thickness**

- Whether the surface-to-surface or node-to-surface constraint formulation (discussed below) is used. If a position tolerance is in effect, a constraint is generated at a slave node for either formulation if the distance between the surfaces, as calculated at the slave node, does not exceed d_{tol}. Additional slave nodes may be tied if the surface-to-surface constraint formulation is used along with an element-based slave surface and a master surface that is not node-based, because the following addendum to the position tolerance criterion applies in such cases: if the distance between the surfaces is within d_{tol} over a significant portion of a slave face (or segment in two dimensions) that forms an angle of less than 30° with the master surface, all slave nodes attached to such a face (or segment) are considered to satisfy the position tolerance.
- The types of surfaces involved (element-based, node-based, or analytical).

Position tolerance for an element-based master surface

The default position tolerance for element-based master surfaces is 5% or 10% of the typical master facet diagonal length for the node-to-surface and surface-to-surface tie formulations, respectively. When using an element-based master surface, the distance between surfaces for a particular point on a slave surface is based on the closest point on the master surface (which may be on the edge of the master surface or within a facet). Figure 33.3.1–2 shows an example with no thickness: nodes 2–14 satisfy the position tolerance criterion for the node-to-surface and surface-to-surface constraint formulations. Significant portions of the end slave segments (that is, the segment connecting nodes 1 and 2 and the

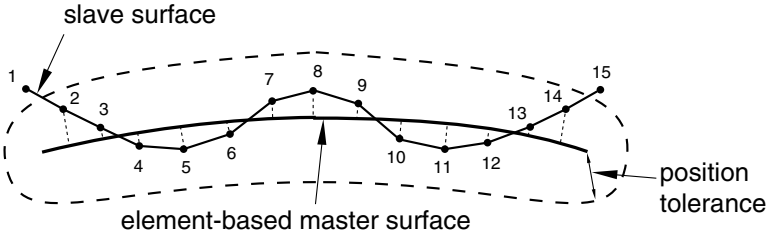


Figure 33.3.1–2 Tolerance region around an element-based master surface with no thickness.

segment connecting nodes 14 and 15) are within the position tolerance shown, so nodes 1 and 15 would also satisfy the position tolerance criterion for the surface-to-surface constraint formulation except for the fact that the angle between the slave and master surfaces is slightly greater than 30° at those locations.

Position tolerance for a node-based master surface

The default position tolerance for a node-based master surface is based on the average distance between nodes in the master surface. The distance between the surfaces for a particular slave node is based on the closest master node. If this distance is less than the position tolerance, Abaqus will create a tie constraint between the slave node, the closest master node, and other master nodes in similar proximity to the slave node. For mismatched meshes across a tied interface, the distance between slave and master nodes can be much larger than the "normal" distance between the surfaces, which can lead to confusion when using a position tolerance criterion with a node-based master surface. Figure 33.3.1–3 shows how the tolerance region is defined around a node-based master surface. The surface-to-surface constraint formulation reverts to the node-to-surface constraint formulation for a node-based master surface.

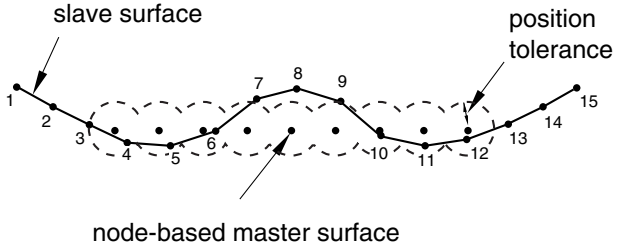


Figure 33.3.1–3 Tolerance region around a node-based master surface with no thickness.

Position tolerance for an analytical rigid master surface

The default position tolerance for tie constraints between an element-based slave surface and an analytical rigid master surface is 5% or 10% of the typical slave facet diagonal length for the node-to-surface and surface-to-surface tied formulations, respectively. The default position tolerance for tie constraints between a node-based slave surface and an analytical rigid master surface is 5% of the typical distance

between slave nodes. When using an analytical rigid master surface, the distance between surfaces for a particular point on the slave surface is based on the closest point on the master surface.

Specifying the constrained nodes directly

This method allows you direct control over which slave nodes are tied.

Input File Usage: *TIE, TIED NSET=*node_set_label*

Abaqus/CAE Usage: Specifying the constrained nodes directly is not supported in Abaqus/CAE.

Unconstrained nodes in tie constraint pairs

Abaqus does not constrain slave nodes to the master surface unless they are included in the tied node set or within the tolerance distance from the master surface at the start of the analysis, as discussed above. Any slave nodes not satisfying these criteria will remain unconstrained for the duration of the simulation; they will never interact with the master surface as part of the tie constraint. In mechanical simulations an unconstrained slave node can penetrate the master surface freely unless contact is defined between the slave node and master surface. The general contact algorithm in Abaqus/Explicit will generate contact exclusions automatically for slave node–master surface combinations corresponding to constrained nodes of tie constraint pairs, but no such contact exclusions are generated for nodes outside the position tolerance of the constraints. In a thermal, acoustic, electrical, or pore pressure simulation an unconstrained slave node will not exchange heat, fluid pressure, electrical current, or pore fluid pressure with the master surface.

Determining which slave nodes have been tied and which slave nodes have not been tied

For each tie constraint pair, Abaqus creates a node set comprising slave nodes that will be tied and a node set comprising slave nodes that will be left unconstrained. These node sets are available for display during postprocessing in Abaqus/CAE, where they are listed as internal node sets.

In addition, Abaqus prints a table in the data (`.dat`) file listing each slave node and the master surface nodes to which it will be tied if model definition data are requested (see "Controlling the amount of analysis input file processor information written to the data file" in "Output," Section 4.1.1). If a constraint cannot be formed for a given slave node, Abaqus/Standard will issue a warning message in the data file.

When creating a model with surface-based tie constraints, it is important to use the information provided by Abaqus to identify any unconstrained nodes and to make any necessary modifications to the model to constrain them.

Constraining the rotational degrees of freedom

By default, Abaqus will constrain the rotational degrees of freedom when they exist on both slave and master surfaces (see Figure 33.3.1–4). You can specify that the rotational degrees of freedom should not be tied.

Input File Usage: *TIE, NO ROTATION

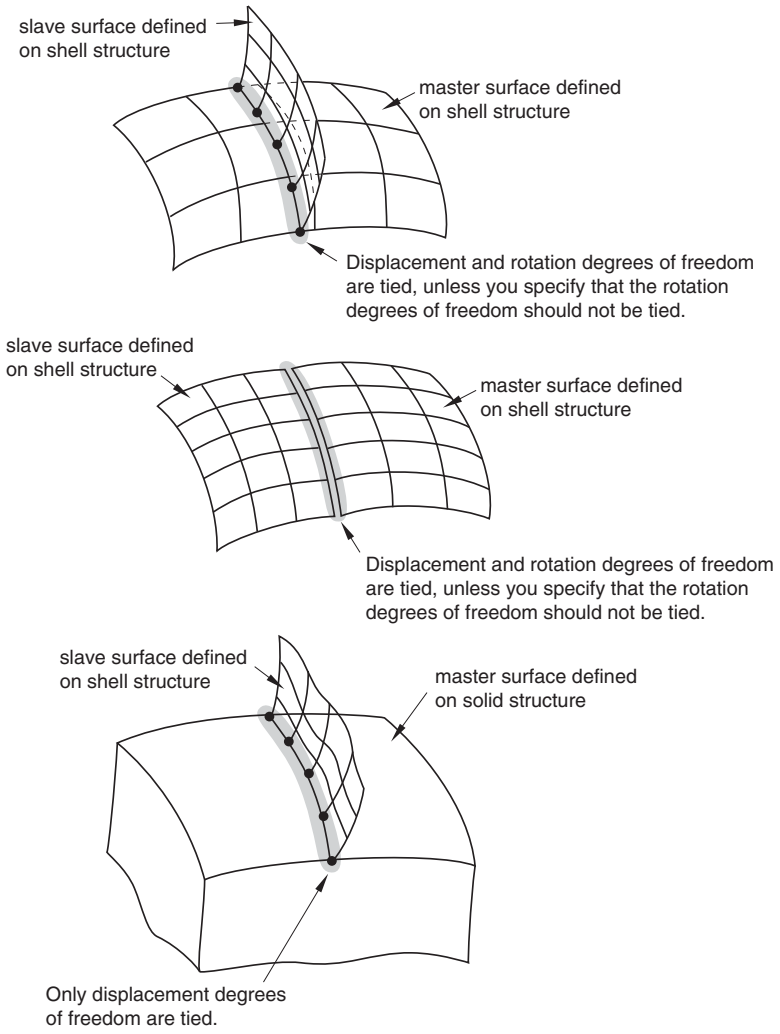


Figure 33.3.1–4 Surface-based tie algorithm.

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Tie**: toggle off **Tie rotational DOFs if applicable**

Constraining the faces of a cyclic symmetric structure in Abaqus/Standard

You can enforce proper constraints on the faces bounding a repetitive sector of a cyclic symmetric structure (see "Analysis of models that exhibit cyclic symmetry," Section 10.4.3). This makes it possible to define a single sector of the cyclic symmetry model together with its axis of cyclic symmetry

to define the behavior of the 360° model. Cyclic symmetry models can be used within the following procedures: static; quasi-static; eigenfrequency extraction, based on the Lanczos solver technique; steady-state dynamics, based on modal superposition; and heat transfer. If an eigenfrequency extraction is performed on a cyclic symmetric model, the nodes involved in the cyclic symmetry constraint cannot be used in any other constraint (e.g., multi-point constraints, equations, rigid bodies, couplings, or kinematic couplings).

Input File Usage: *TIE, CYCLIC SYMMETRY

This parameter can be used only with the *CYCLIC SYMMETRY MODEL option.

Abaqus/CAE Usage: Interaction module: **Interaction→Create**: **Cyclic symmetry**

The surface-based tie constraint formulation

Abaqus uses the criteria discussed above to determine which slave nodes will be tied to the master surface. Abaqus then forms constraints between these slave nodes and the nodes on the master surface. A key aspect in forming the constraint for each slave node is determining the tie coefficients. These coefficients are used to interpolate quantities from the master nodes to the tie point. Abaqus can use one of two approaches to generate the coefficients: the "surface-to-surface" approach or the "node-to-surface" approach.

If an analysis carried out with Abaqus/Standard is imported into Abaqus/Explicit or vice-versa, the tie constraints are not imported and must be redefined. If the imported analysis is essentially a continuation of the original analysis, it is important that the tie constraints are as similar as possible. Hence, you should make sure that the same constraint type is used. If the default approach was used in the original Abaqus/Standard analysis, the surface-to-surface approach should be specified in the Abaqus/Explicit analysis. Similarly, if the default approach was used in the original Abaqus/Explicit analysis, the node-to-surface approach should be specified in the Abaqus/Standard analysis.

The "surface-to-surface" approach

The "surface-to-surface" approach minimizes numerical noise for tied interfaces involving mismatched meshes. The surface-to-surface approach enforces constraints in an average sense over a finite region, rather at discrete points as in the traditional node-to-surface approach. The surface-to-surface formulation for surface-based tie constraints is similar to the surface-to-surface contact formulation (see "Contact formulations in Abaqus/Standard," Section 36.1.1); however, a fundamental difference is that each surface-based tie constraint involves only one slave node (and multiple master nodes), whereas each surface-to-surface contact constraint involves multiple slave nodes.

The surface-to-surface approach is used by default in Abaqus/Standard with exceptions noted below, and it is optional in Abaqus/Explicit. For the case of infinite acoustic elements tied to shell elements in Abaqus/Standard the added cost of the surface-to-surface approach can be quite significant; therefore, the node-to-surface approach is used by default in this case. If the surface-to-surface approach is "on by default" or explicitly specified, Abaqus automatically reverts to the node-to-surface approach for individual tie constraints in the following circumstances:

- if either of the surfaces being tied is node-based;
- if the projection along the slave surface normal direction does not intersect the master surface; or
- if single-sided slave and master surfaces have surface normals in approximately the same direction.

Abaqus/Explicit may automatically add a small amount of artificial mass to the model to maintain numerical stability if the surface-to-surface approach is specified.

The surface-to-surface approach generally involves more master nodes per constraint than the node-to-surface approach, which tends to increase the solver bandwidth in Abaqus/Standard and, therefore, can 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 the surface-to-surface approach being quite costly:

- A large fraction of tied nodes (degrees of freedom) in the model
- The master surface being more refined than the slave surface
- Multiple layers of tied shells, such that the master surface of one tie constraint acts as the slave surface of another tie constraint

Input File Usage: *TIE, TYPE=SURFACE TO SURFACE

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Tie**: **Discretization method: Surface to surface**

The "node-to-surface" approach

The traditional "node-to-surface" approach (which is used by default in Abaqus/Explicit and is optional in Abaqus/Standard) sets the coefficients equal to the interpolation functions at the point where the slave node projects onto the master surface. This approach is somewhat more efficient and robust for complex surfaces.

For the node-to-surface method of establishing the tie coefficients with an element-based master surface, the point on the surface closest to each slave node is calculated and used to determine the master nodes that are going to form the constraint (see Figure 33.3.1–5). For example, nodes 202, 203, 302, and 303 are used to constrain node *a*; nodes 204 and 304 are used to constrain node *b*; and node 402 is used to constrain node *c*.

Input File Usage: *TIE, TYPE=NODE TO SURFACE

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Tie**: **Discretization method: Node to surface**

Choosing the slave and master surfaces of a surface-based tie constraint

The choice of slave and master surfaces can have a significant effect on the accuracy of the solution, in particular if the "node-to-surface" approach is used. The effect is much less (and the accuracy generally better) for the "surface-to-surface" approach. In either case, if both surfaces in a constraint pair are deformable surfaces, the master surface should be chosen as the surface with the coarser mesh for best accuracy.

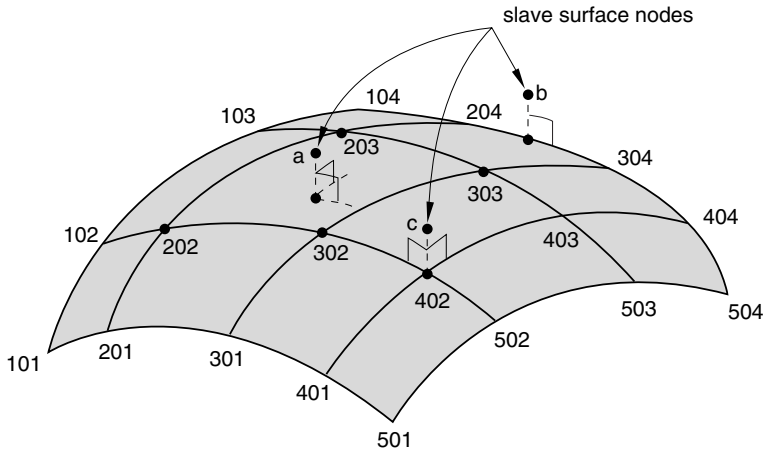


Figure 33.3.1–5 Searching for the points on an element-based master surface that are closest to nodes *a*, *b*, and *c*.

In Abaqus/Standard a rigid surface cannot act as a slave surface in a tie constraint. To comply with this rule, the capability to automatically resolve overconstraints in Abaqus/Standard (see "Overconstraint checks," Section 33.6.1) will modify tie constraint definitions in the following cases:

- Tie constraints between two surfaces of the same rigid body are removed.
- Tie constraints between two surfaces of two rigid bodies are replaced by a BEAM-type connector between the respective rigid body reference nodes.
- Tie constraints specified with a purely rigid slave surface and a purely deformable master surface are modified to reverse the master and slave assignments unless this is not possible due to other modeling restrictions (in which case an error message is issued).

These methods are not applied if the slave surface that you specified is partially rigid and partially deformable; Abaqus/Standard issues an error message in such cases.

In acoustic, structural-acoustic, and elastic wave propagation problems care should be exercised when tying meshes of highly dissimilar refinement. If two media have different wave speeds, the optimal meshes for each of the media will have different characteristic element lengths: the faster medium will have larger elements. If surfaces of these meshes are used in a tie constraint, the surface of the finer mesh (of the slower medium) should be designated as the slave. Nevertheless, in the region near the tied surfaces, the physical wave phenomena in *both* fast and slow media will typically have length scales characteristic of the slower medium; that is, of the shortest length scale in the physical problem. Therefore, if these phenomena are important, the mesh of the faster medium should be refined to the scale of the slower medium in the vicinity of the contact region.

Adjusting the surfaces and considering offsets

By default, with the exceptions mentioned below, Abaqus will automatically reposition the slave nodes to be tied in the initial configuration without causing strain to resolve gaps such that the surfaces are just touching, accounting for any shell thickness (unless you have specified that thickness should not be accounted for, as discussed above in the context of the position tolerance criterion) but not accounting for beam or membrane thickness. One exception is that no adjustments are made where tied surfaces are closer together than the combined half-shell thickness. All adjustments are performed such that the slave and master surfaces are never pushed apart; that is, the reference surfaces will only become closer as a result of the adjustments.

It is recommended that you allow the automatic adjustments to occur, especially if neither surface has rotations; in this case a constant offset vector is used, so incorrect behavior of the constraint under rigid body rotation can occur when slave nodes are not lying exactly on the master surface. Adjustments are not made if the slave surface belongs to a substructure or when either the slave or master surface is a beam element-based surface; in the latter cases you should locate the beam element nodes with the desired offset from the other surface.

Input File Usage: *TIE, ADJUST=YES *or* NO

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Tie**: toggle **Adjust slave node initial position**

Criteria for adjustment

A slave node is considered for adjustment if both of the following conditions are met:

- The slave node satisfies whatever criterion is in effect for generating a constraint (either because it satisfies the position tolerance criterion or belongs to the specified node set of constrained slave nodes, as previously discussed).
- The slave node is more than the combined thickness of the slave and master surfaces away from its projection point on the master reference surface, accounting for any offset of the element reference surfaces from the respective element midsurfaces.

For an element-based master surface a slave node will be moved toward the closest point on the master surface; for a node-based master surface a slave node will be moved toward the closest master node. The corrected position of an adjusted slave node is determined from the combined effects of shell element thickness and any specified reference surface offset relative to the shell midsurface of either slave or master surfaces. Figure 33.3.1–6 shows the adjusted slave node position in an example with two shell element-based surfaces tied together (in this example one of the element reference surfaces is offset from the element midsurface). It is assumed that the surfaces were farther apart than shown in Figure 33.3.1–6 prior to the adjustment; otherwise, the slave nodes would not have been adjusted.

Adjustments are made only for slave nodes that are included in the user-specified tied node set or that meet the tolerance criteria described above.

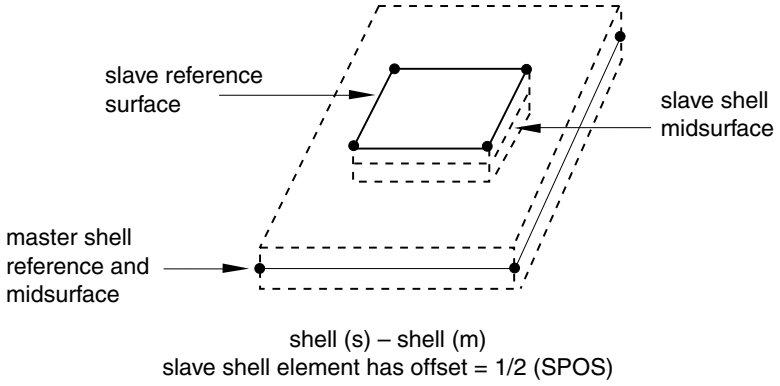


Figure 33.3.1–6 Adjusted slave node position for two shell element-based surfaces tied together. The slave shell element has an offset of 0.5.

Adjustments for overlapping constraints

Nodal adjustments for tie constraints are processed sequentially in the order of the constraint definitions 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 are less likely to occur in Abaqus/Explicit because the adjustments in Abaqus/Explicit are automatically processed in the chaining order discussed in "Overlapping constraints." These conflicts can be avoided in Abaqus/Standard 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 slave nodes and later as master 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.

Accounting for an offset between tied surfaces

Abaqus allows a gap to exist between tied surfaces. Such gaps may exist if you prevent nodal adjustments for tied surfaces. A gap between the reference surfaces may remain due to the presence of shell thickness even if nodal adjustments are performed. Figure 33.3.1–7 shows some cases where an offset between the reference surfaces may be desirable for tied surface pairs to account for shell or beam thickness. Rigid body motion is properly accounted for when the nodes are separated by a finite distance when at least one of the surfaces is based on shell or beam elements; when the master surface is an analytical

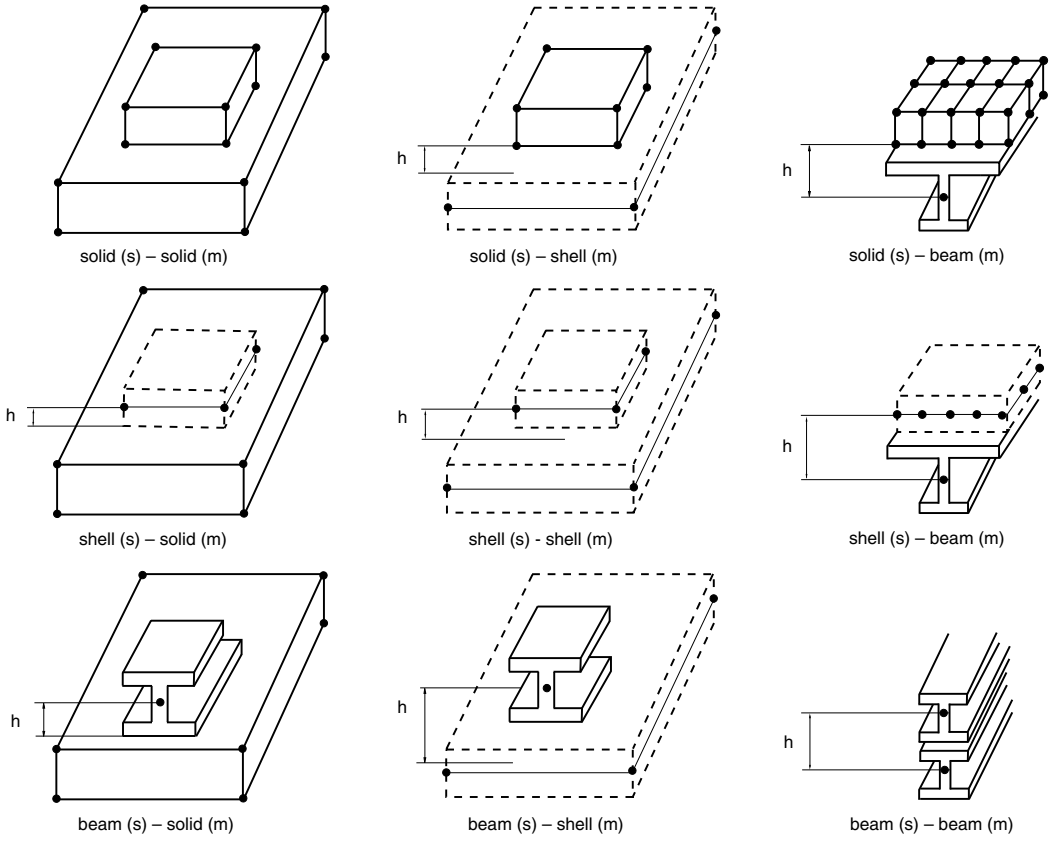


Figure 33.3.1–7 Tie constraints being applied between surfaces based on various element types (h = offset between slave and master surfaces).

rigid surface; or, in the case of node-based surfaces, when the nodes on at least one surface have active rotational degrees of freedom.

The nature of the constraint on translational motion between surfaces in Abaqus depends on whether there is an offset between the surfaces and on which surfaces have rotational degrees of freedom, as discussed below.

Neither surface has rotational degrees of freedom

If neither surface has rotational degrees of freedom, the global translational degrees of freedom of the slave node and the closest point on the master surface are constrained to be the same. When an offset exists, Abaqus will enforce the constraint through the fixed offset like a PIN-type MPC when the nodes in the MPC are not coincident. Because the fixed offset does not rotate, the surface-based constraint will not represent rigid body rotation correctly. The constraint will represent rigid body motion correctly

when the offset is zero. This behavior can be ensured by specifying that all tied slave nodes should be moved onto the master surface.

Only one surface has rotational degrees of freedom

If the slave surface has rotational degrees of freedom and the master surface does not, the translational motion is constrained at the closest point on the master reference surface. When the reference surfaces are offset, a moment will be applied to each slave node based on the constraint force times the offset distance. Similarly, if the master surface has rotational degrees of freedom and the slave surface does not, the translational motion is constrained at each slave node and a moment will be applied to the relevant nodes on the master surface if an offset exists. In either case the surface-based constraint will behave correctly under rigid body rotation regardless of the amount of offset.

Both surfaces have rotational degrees of freedom

If both surfaces have rotational degrees of freedom, are not offset, and the rotations are tied, each slave node is constrained to the master surface like a TIE-type MPC. If an offset exists between the surfaces, the constraint acts like a BEAM-type MPC between the slave node and the closest point on the master reference surface.

If the rotations are not tied, Abaqus allows you to choose the location of the translational constraint. It can be enforced at the master reference surface, the slave reference surface, or anywhere in between. The location of the translational constraint enforcement for surfaces where the rotations are not tied will affect the distribution of moment to each of the surfaces. The most physically reasonable choice is to locate the constraint at the point where the actual top or bottom sides of each surface meet. The constraint then models a perfect adhesive between the surfaces, which transfers shear stress to each surface. Abaqus will choose the location of the translational constraint as follows:

- If the master surface is shell element-based, the translational constraint is enforced on the top or bottom side of the master surface.
- If the slave surface is shell element-based and the master surface is not, the translational constraint is enforced at the top or bottom side of the slave surface.
- Otherwise, the translational constraint is enforced at the master reference surface.

To override these default locations, you can specify a constraint ratio for the tie constraint equal to the fractional distance between the master reference surface and the slave node at which the translational constraint should act. Figure 33.3.1–8 shows an example of the use of a constraint ratio to prescribe the location of the translational constraint between two shell surfaces that are tied together with no rotational constraints. The distance between the master reference surface and the slave reference surface is b. The prescribed constraint ratio, r, is then used to locate the translational constraint at a distance a from the master reference surface. All distances are measured along the vector between the slave node and its projection point onto the master reference surface. The constraint behavior is then similar to that of two rigid beams pinned together, as shown.

Input File Usage: *TIE, CONSTRAINT RATIO=*value*

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Tie**: **Constraint ratio**

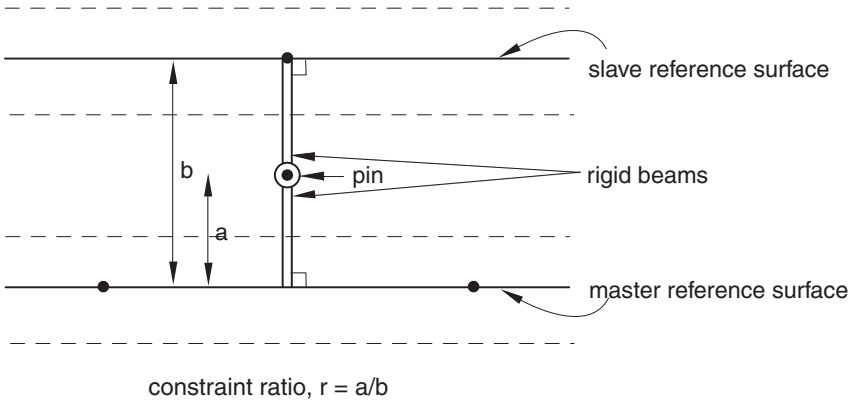


Figure 33.3.1–8 Use of a constraint ratio to prescribe the location of the translational constraint.

Constraining a surface to a three-dimensional beam

The master surface for a tie constraint can be based on three-dimensional beam elements. For this case each slave node is projected onto the line formed by the nodes of the beam elements in the undeformed configuration to find the projection point. During the subsequent analysis the motion of each slave node is rigidly constrained to the motion (translation and rotation) of its projection point; i.e., each slave node and its projection point are connected by a rigid beam. Constraining other elements to a beam element-based master surface allows modeling of interactions between the surface of a (complex) beam section and its surroundings, without having to model the beam with continuum and/or shell elements. This feature can be particularly useful for modeling acoustic-structural interactions.

Note: Abaqus/CAE currently does not support master surfaces based on beam elements.

Use of tie constraints in non-mechanical simulations

The surface-based tie constraint capability can be used in models where the nodal degrees of freedom on both the slave and master surfaces include electrical potential, pore pressure, acoustic pressure, and/or temperature. Except for the type of nodal degree of freedom being constrained, Abaqus uses exactly the same formulation for the tie constraint in nonmechanical simulations as it does for mechanical simulations. In general, degrees of freedom common to both surfaces are tied, and any other degrees of freedom are unconstrained.

The case of structural-acoustic constraints is the exception to this rule. Here, appropriate relations between the acoustic pressure on the fluid surface and displacements on the solid surface are formed internally (see "Acoustic, shock, and coupled acoustic-structural analysis," Section 6.10.1). The displacements and/or pressure degrees of freedom on the surfaces are the only ones affected; rotations are ignored by the tie constraint in this case.

The internally computed structural-acoustic coupling conditions use surface areas and normal directions associated with the slave surface elements. The slave surface for structural-acoustic tie

constraints cannot be a node-based surface. In two-dimensional analyses the out-of-plane thickness of the slave elements is required. Generally, this thickness is the thickness specified on the section definition for the slave surface elements. However, when beam elements form the slave surface in a tie constraint pair with acoustic elements, a unit thickness in the out-of-plane direction is assumed for the beams.

In Abaqus/Standard you can define coupling between solid medium and acoustic medium infinite elements along the surfaces that extend to infinity. These surfaces are defined using the edges of the acoustic elements and sides numbered "2" and higher of the solid medium infinite elements. The infinite surfaces of solid medium and acoustic infinite elements can be coupled only through the use of a surface-based tie constraint. As shown in Figure 33.3.1–9, the acoustic infinite elements must be the slave elements and the edges of the acoustic infinite elements should lie within the specified position tolerance to the solid medium infinite element base facets.

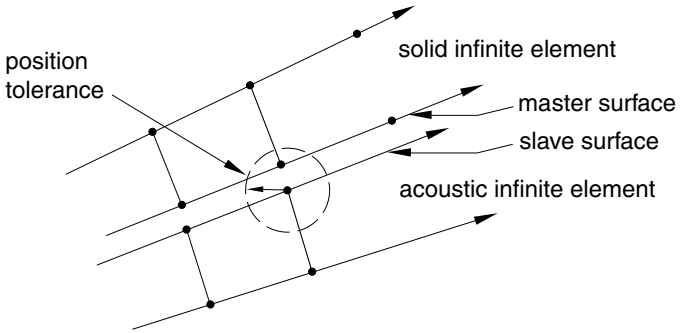


Figure 33.3.1–9 Use of a surface-based tie constraint to prescribe the coupling between solid medium and acoustic medium infinite elements.

If the base facets of acoustic infinite elements are to be coupled to solid medium finite elements, to solid medium infinite elements, or to structural elements, either a surface-based tie constraint or acoustic-structural interaction elements can be used. Surfaces defined on solid medium infinite elements cannot be used in a surface-based tie constraint in Abaqus/Explicit.

Table 33.3.1–3 enumerates all possible cases. For other slave-master pairings not listed in this table, an error message will be issued.

Table 33.3.1–3 Possible slave-master surface pairings.

Slave Surface	Master Surface	Degrees of Freedom Tied
Acoustic	Acoustic	Acoustic pressure
Acoustic	Stress	Translations
Stress	Acoustic	Acoustic pressure
Stress	Stress	Translations and/or rotations

Slave Surface	Master Surface	Degrees of Freedom Tied
Heat-Stress	Stress	Translations and/or rotations
Stress	Heat-Stress	Translations and/or rotations
Heat-Stress	Heat-Stress	Temperature, translations and/or rotations
The following surface pairings are available only in Abaqus/Standard:		
Heat transfer	Heat transfer	Temperature
Electrical-Heat	Heat transfer	Temperature
Heat transfer	Electrical-Heat	Temperature
Electrical-Heat	Electrical-Heat	Temperature and electric potential
Pore-Stress	Pore-Stress	Pore pressure and translations
Pore-Stress	Stress	Translations
Stress	Pore-Stress	Translations

Tie constraints versus tied contact in Abaqus/Standard

There are the following advantages to using a surface-based tie constraint in Abaqus/Standard instead of defining tied contact as discussed in "Defining tied contact in Abaqus/Standard," Section 34.3.7:

- Degrees of freedom of the slave surface nodes will be eliminated.
- The tie constraint is more efficient in terms of the size of the fronts of the operator matrix because fewer master surface nodes are associated with each slave node.
- Rotational degrees of freedom as well as translational degrees of freedom can be tied.
- Tie constraints are much more general since they allow the use of general surfaces.
- Surface offsets and shell thickness are taken into account.

Overlapping constraints

In a model with multiple tie constraint definitions it is possible that the slave and master surfaces of different tie constraint definitions may intersect. If two tie constraint definitions have part or all of their master surfaces in common or if the surfaces tied are layered (i.e., the master surface of one tie constraint definition acts as the slave surface of a subsequent tie constraint definition), Abaqus will attempt to chain the constraint definitions together. This will reduce the number of degrees of freedom and lower the computational expense, resulting in faster run times. However, in a model with multiple tie constraint definitions if nodes on the slave surface of one tie constraint definition are part of the slave surface of other tie constraint definitions, an overconstraint occurs. In most cases the overconstraint is due to the existence of redundant constraints, and it is safe to eliminate this redundancy. However, the overconstraint may also be due to conflicting constraints, in which case the problem is due to a modeling error that you should correct. Simulation results will vary depending on which constraint is removed to

avoid an overconstraint if the overlapping constraints are not identical. It is recommended that, wherever possible, you order the slave and master surfaces of the constraint definitions to avoid intersecting slave surfaces. See "Adjustments for overlapping constraints" for a discussion of initial strain-free adjustments for overlapping constraints.

Overconstrained slave nodes in Abaqus/Standard

If an overconstraint occurs, Abaqus/Standard issues an error message unless the constraints are redundant or nearly redundant, as discussed below. As discussed previously, each tie constraint involves a single slave node and a set of master nodes with nonzero tie coefficents. Abaqus/Standard considers tie constraints involving the same slave node to be nearly redundant if at least one node is common among the respective sets of master nodes with nonzero tie coefficients. In such cases, rather than issuing an error message, Abaqus/Standard issues a warning message and only enforces one of the constraints.

The surface-based tie constraint is imposed in Abaqus/Standard by eliminating the degrees of freedom on the slave surface; therefore, nodes on the slave surface should not be used to apply boundary conditions, nor should they be used in any subsequent tie, multi-point, equation, or kinematic coupling constraint (see "Overconstraint checks," Section 33.6.1, for a more complete discussion of overconstraints in Abaqus/Standard).

Overconstrained slave nodes in Abaqus/Explicit

In contrast, Abaqus/Explicit treats overconstraints with a penalty method, thus enforcing the constraints in an average sense; the computational cost of the analysis may increase in these cases.

In addition, if the slave surface for a tie constraint definition in Abaqus/Explicit is part of a rigid body while the master surface comprises a deformable element- or node-based surface and the master surface acts as the slave surface in a subsequent tie constraint definition, the resolution of the resulting constraints can prove to be computationally intensive. It is recommended that, wherever possible, you order the slave and master surfaces of the constraint definitions to avoid such a situation.

Limitations

The following limitations exist for tie constraints:

- Surface-based tie constraints cannot be used to connect gasket elements that model thickness-direction behavior only.
- A rigid surface cannot act as a slave surface in a constraint pair in Abaqus/Standard.
- A slave node of a tie constraint cannot act as a slave node of another constraint in Abaqus/Standard.
- Tie constraints cannot be used to tie infinite elements to finite elements in Abaqus/Explicit. To couple infinite and finite elements in Abaqus/Explicit, the elements must share nodes.
- The axisymmetric solid Fourier elements with nonlinear, asymmetric deformation cannot form element-based surfaces; therefore, such surfaces cannot be used in tie constraints.

33.3.2 COUPLING CONSTRAINTS

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Surfaces: overview," Section 2.3.1
- *COUPLING
- *KINEMATIC
- *DISTRIBUTING
- "Defining coupling constraints," Section 15.15.4 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

The surface-based coupling constraint:

- couples the motion of a collection of nodes on a surface to the motion of a reference node;
- is of type kinematic when the group of nodes is coupled to the rigid body motion defined by the reference node;
- is of type distributing when the group of nodes can be constrained to the rigid body motion defined by a reference node in an average sense by allowing control over the transmission of forces through weight factors specified at the coupling nodes;
- automatically selects the coupling nodes located on a surface lying within a region of influence;
- can be used with two- or three-dimensional stress/displacement elements; and
- can be used in geometrically linear and nonlinear analysis.

Surface-based coupling definitions

The surface-based coupling constraint in Abaqus provides coupling between a reference node and a group of nodes referred to as the "coupling nodes." This option provides the same functionality as the kinematic coupling constraint and the distributing coupling elements (DCOUP2D, DCOUP3D) in Abaqus/Standard with a surface-based user interface. The coupling nodes are selected automatically by specifying a surface and an optional influence region. The procedure used to define the coupling nodes is discussed below.

For a distributing coupling constraint, the distributing weight factors are calculated automatically if the surface is an element-based surface. In such a case the weight factors are based on the tributary area at each coupling node, except for a surface along a shell edge, where the weight factors are based on the tributary edge length. Furthermore, the distributing weight factors can be modified using one of several weighting methods, which allow the forces transferred to the coupling nodes to vary inversely with the radial distance from the reference node.

Typical applications

The coupling constraint is useful when a group of coupling nodes is constrained to the rigid body motion of a single node. The coupling constraint can be employed effectively in the following applications:

- To apply loads or boundary conditions to a model. Figure 33.3.2–1 illustrates the use of a kinematic coupling constraint to prescribe a twisting motion to a model without constraining the radial motion.

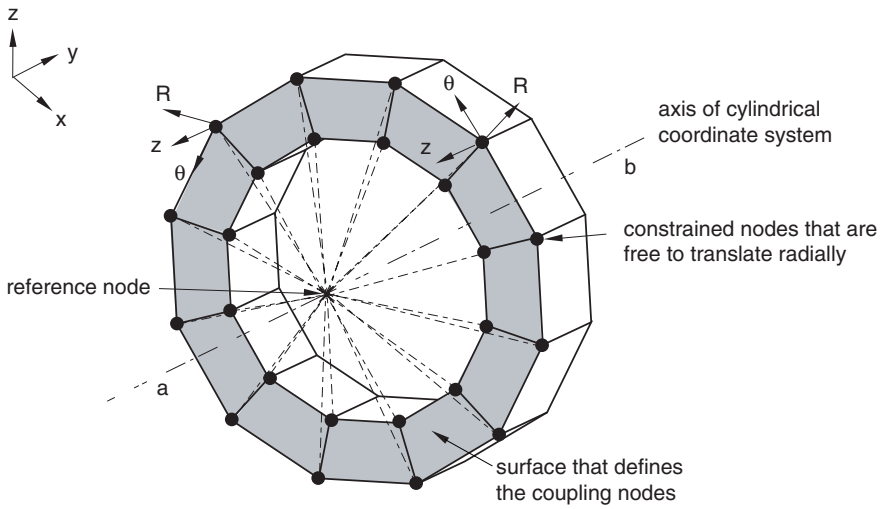


Figure 33.3.2–1 Kinematic coupling constraint.

Figure 33.3.2–2 illustrates a distributing coupling constraint used to prescribe a displacement and rotation condition on a boundary where relative motion between the nodes on the boundary is required. In this example a twist is prescribed at the end of the structure that is expected to warp and/or deform within the end surface.

- To distribute loads on a model, where the load distribution can be described with a moment-of-inertia expression. Examples of such cases include the classic bolt-pattern and weld-pattern distribution expressions.
- To apply dimensionality transitions between continuum and structural elements. For example, a distributing coupling allows flexible coupling between structural and solid elements.
- To model end conditions. For example, modeling a rigid end plate or modeling plane sections of a solid to remain planar can be done easily with a kinematic coupling definition.
- To simplify modeling of complex constraints. In a kinematic coupling definition the degrees of freedom that participate in the constraint may be selected individually in a local coordinate system.
- To model interactions with other constraints, such as connector elements. For example, a hinged part may be modeled more realistically by two distributing coupling definitions, whose reference nodes

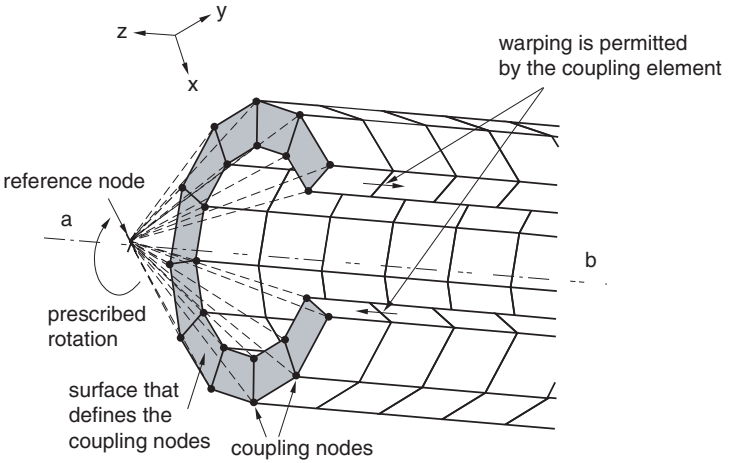


Figure 33.3.2–2 Distributing coupling constraint.

are connected by a hinge connector element. The load transfer then occurs between two "clouds" of nodes, rather than between two single nodes. "Substructure analysis of a one-piston engine model," Section 4.1.10 of the Abaqus Example Problems Manual, illustrates this use of connector elements in conjunction with coupling constraints to model a one-piston engine.

Defining the coupling constraint

Defining a coupling constraint requires the specification of the reference node (also called the constraint control point), the coupling nodes, and the constraint type. The coupling constraint associates the reference node with the coupling nodes. A name must be assigned to the constraint and may be used in postprocessing with Abaqus/CAE. A node number or node set name may be specified for the reference node. If a node set is specified, the node set must contain exactly one node. The reference node for a kinematic coupling constraint has both translational and rotational degrees of freedom. The surface on which the coupling nodes are located can be node-based; element-based; or, in Abaqus/Explicit, a combination of both surface types. You can specify an optional radius of influence that limits the coupling nodes to a specific region on the surface. Details on how coupling nodes are defined by specifying an influence region are discussed below.

The constraint type can be either kinematic or distributing, as discussed below.

Input File Usage: Use the following options:

∗COUPLING, CONSTRAINT NAME=*name*, REF NODE=*n*, SURFACE=*surface*
∗KINEMATIC *or* ∗DISTRIBUTING

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Coupling**: **Coupling type**: **Kinematic** or **Distributing**

Specifying a region of influence

By default, coupling nodes belonging to the entire surface are selected for the coupling definition. You can limit the coupling nodes to lie within a spherical region centered about the reference node by defining a radius of influence.

The procedure by which coupling nodes are selected for the constraint definition depends on the surface type:

- For a node-based surface, all the nodes defined by the surface definition that fall within the influence region are selected for the coupling definitions.
- For an element-based surface, the surface facets that are either fully or partially inscribed by the influence region are determined. All nodes belonging to these facets, whether or not these nodes fall within the influence region, are selected for the coupling nodes. When the influence radius is less than the distance to the closest coupling node, Abaqus selects all nodes belonging to the closest facet. If the projection of the reference node on the surface falls on an edge or a vertex of multiple facets, all nodes belonging to these facets adjoining the edge or vertex are included in the coupling definition. In the case where the influence radius is less than the distance to the closest coupling node, adjacent surface faces must have consistent normal directions; otherwise, Abaqus issues an error message.
- A distributing coupling constraint must include at least two coupling nodes. If fewer than two coupling nodes are found, Abaqus issues an error message during input file preprocessing.

Input File Usage: ∗COUPLING, CONSTRAINT NAME=*name*, REF NODE=*n*, SURFACE=*surface*, INFLUENCE RADIUS=*r*

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Coupling**: **Influence radius**: **Specify**

Kinematic coupling constraints

Kinematic coupling constrains the motion of the coupling nodes to the rigid body motion of the reference node. The constraint can be applied to user-specified degrees of freedom at the coupling nodes with respect to the global or a local coordinate system.

Kinematic constraints are imposed by eliminating degrees of freedom at the coupling nodes. In Abaqus/Standard once any combination of displacement degrees of freedom at a coupling node is constrained, additional displacement constraints—such as MPCs, boundary conditions, or other kinematic coupling definitions—cannot be applied to any coupling node involved in a kinematic coupling constraint. The same limitation applies for rotational degrees of freedom. This restriction does not apply in Abaqus/Explicit. See “Kinematic constraints: overview,” Section 33.1.1, for more information.

Input File Usage: Use both of the following options to define a kinematic coupling constraint:

∗COUPLING
∗KINEMATIC
first dof, *last dof*

For example, the following coupling constraint is used to constrain degrees of freedom 1, 2, and 6 on surface **surfA** to reference node 1000:

```
*COUPLING, CONSTRAINT NAME=C1, REF NODE=1000,
SURFACE=surfA
*KINEMATIC
1, 2
6,
```

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Coupling**: **Coupling type**: **Kinematic**: toggle on the degrees of freedom

Translational degrees of freedom

Translational degrees of freedom are constrained by eliminating the specified degrees of freedom at the coupling nodes. When all translational degrees of freedom are specified, the coupling nodes follow the rigid body motion of the reference node.

Rotational degrees of freedom

Rotational degrees of freedom are constrained by eliminating the specified degrees of freedom at the coupling nodes.

All combinations of selected rotational degrees of freedom result in rotational behavior identical to existing MPC types:

- Selection of three rotational degrees of freedom along with three displacement degrees of freedom is equivalent to MPC type BEAM.
- Selection of two rotational degrees of freedom is equivalent to MPC type REVOLUTE in Abaqus/Standard.
- Selection of one rotational degree of freedom is equivalent to MPC type UNIVERSAL in Abaqus/Standard.

In Abaqus/Standard internal nodes are created by the kinematic coupling to enforce the constraints that are equivalent to MPC types REVOLUTE and UNIVERSAL. These nodes have the same degrees of freedom as the additional nodes used in these MPC types and are included in the residual check for nonlinear analysis.

Specifying a local coordinate system

The kinematic coupling constraint can be specified with respect to a local coordinate system instead of the global coordinate system (see "Orientations," Section 2.2.5). Figure 33.3.2–1 illustrates the use of a local coordinate system to constrain all but the radial translation degrees of freedom of the coupling nodes to the reference node. In this example a local cylindrical coordinate system is defined that has its axis coincident with the structure's axis. The coupling node constraints are then specified in this local coordinate system.

Input File Usage: *COUPLING, ORIENTATION=*local*

For example, the following input is used to specify the kinematic coupling constraint shown in Figure 33.3.2–1:

```
*ORIENTATION, SYSTEM=CYLINDRICAL, NAME=COUPLEAXIS
0.0, -1.0, 0.0, 0.0, 1.0, 0.0
*COUPLING, REF NODE=500, SURFACE=Endcap,
ORIENTATION=COUPLEAXIS
*KINEMATIC
2, 3
```

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Coupling**: **Edit**: select local coordinate system

Constraint direction and finite rotation

In geometrically nonlinear analysis steps the coordinate system in which the constrained degrees of freedom are specified will rotate with the reference node regardless of whether the constrained degrees of freedom are specified in the global coordinate system or in a local coordinate system.

Distributing coupling constraints

Distributing coupling constrains the motion of the coupling nodes to the translation and rotation of the reference node. This constraint is enforced in an average sense in a way that enables control of the transmission of loads through weight factors at the coupling nodes. Forces and moments at the reference node are distributed either as a coupling node-force distribution only (default) or as a coupling node-force and moment distribution. The constraint distributes loads such that the resultants of the forces (and moments) at the coupling nodes are equivalent to the forces and moments at the reference node. For cases of more than a few coupling nodes, the distribution of forces/moments is not determined by equilibrium alone, and distributing weight factors are used to define the force distribution.

The moment constraint between the rotation degrees of freedom at the reference node and the average rotation of the cloud nodes can be released in one direction in a two-dimensional analysis and one, two, or three directions in a three-dimensional analysis. In a three-dimensional analysis you can specify the moment constraint directions in the global coordinate system or in a local coordinate system. All available translational degrees of freedom at the reference node are always coupled to the average translation of the coupling nodes.

In a three-dimensional Abaqus/Standard analysis if all three moment constraints are released by specifying only degrees of freedom 1 through 3, only translation degrees of freedom will be activated on the reference node. If only one or two rotation degrees of freedom have been released, all three rotation degrees of freedom are activated at the reference node. In this case you must ensure that proper constraints have been placed on the unconstrained rotation degrees of freedom to avoid numerical singularities. Most often this is accomplished by using boundary conditions or by attaching the reference node to an element such as a beam or shell that will provide rotational stiffness to the unconstrained rotation degrees of freedom.

In Abaqus/Explicit releasing one or more of the moment constraints may lead to significant computational performance degradation. This is also the case when other constraints intersect the cloud of coupling nodes. In these cases, the degradation in performance is particularly noticeable when a large number of such distributed couplings are present in the model or when the size of the constrained "cloud" is large. For that matter, when the modeling conditions mentioned above are encountered, the size of the coupling nodes cloud is limited to 1000. To alleviate the released moment constraint issue, the following modeling technique can be used (also available in Abaqus/Standard): constrain all moments in the distributed coupling and use an appropriate connector element at the reference node (such as REVOLUTE, HINGE, CARDAN or BUSHING) to model released moments at the coupling's reference node. This technique has also the advantage of being able to specify finite compliance such as elasticity, plasticity or damage in the "released" rotational component.

Input File Usage: *DISTRIBUTING
first dof, *last dof*

If no degrees of freedom are specified, all available degrees of freedom are coupled. If you specify one or more rotation degrees of freedom but not all available translation degrees of freedom, Abaqus issues a warning message and adds all available translation degrees of freedom to the constraint.

For example, the following coupling constraint is used to constrain degrees of freedom 1–5 on the reference node 1000 to the average translation and rotation of surface **surfA**:

```
*COUPLING, CONSTRAINT NAME=C1, REF NODE=1000,
SURFACE=surfA
*DISTRIBUTING
1, 5
```

In this example the moment constraint between the reference node and the coupling nodes will be released in the 6-direction but will be enforced in the 4- and 5-directions. This provides a "revolute-like" rotation connection between the reference node and the coupling nodes (see "General multi-point constraints," Section 33.2.2).

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Coupling**: **Coupling type**: **Distributing**: toggle on the rotational degrees of freedom (Abaqus/CAE automatically constrains the translational degrees of freedom)

Node-based surface

User-defined weight factors are used for node-based surfaces. The cross-sectional areas specified in the surface definition are used as the weight factors (see "Node-based surface definition," Section 2.3.3).

Element-based surface

For element-based surfaces the weight factors are calculated by Abaqus. The default weight distribution is based on the tributary surface area at each coupling node, except for a surface along a shell edge

where the weight distribution is based on the tributary edge length. The procedure used to calculate the default weight factors is designed to ensure that if a radius of influence is prescribed, the default weight distribution varies smoothly with the influence radius.

Calculating the default distributing weight factors

The procedure to calculate the distributing weight factors depends on whether or not an influence radius is specified.

- If no influence radius is specified, the entire surface is used in the coupling definition. In this case all nodes located on the surface are included in the coupling definition and the distributing weight factor at each coupling node is equal to the tributary surface area.
- If an influence radius is specified, the default distributing weight factors at the coupling nodes are calculated as follows:
  1. A "participation factor" is calculated for each surface facet. The participation factor is defined below.
  2. The tributary nodal area (or tributary edge length along a shell edge) at each facet node is computed and is multiplied by the facet participation factor.
  3. The coupling node distributing weight factor is computed as the sum of the corresponding facet nodal areas (calculated above) for all joining facets.

Calculating the facet participation factor

The participation factor defines the proportion of the facet's area that contributes to the distributing weight factors when an influence radius is specified. The participation factor varies between zero and one.

To define the participation factor, the distance of the facet node closest to the reference node, r_{min}, and the distance of the facet node farthest from the reference node, r_{max}, are calculated.

- If $r_{max} \leq r_{infl}$, where r_{infl} is the influence radius, all facet nodes lie within the influence region; and a participation factor of one is used.
- If $r_{min} \geq r_{infl}$, none of the facet nodes lie within the influence region; and the participation factor is set to zero.
- If $r_{max} > r_{infl}$, the facet is partially inscribed in the influence region; and the facet is assigned a participation factor equal to $(r_{infl} - r_{min})/(r_{max} - r_{min})$.

If all coupling nodes fall outside the influence radius (i.e., $r_{min} > r_{infl}$ for all facets), Abaqus selects all nodes belonging to the closest facets (as outlined under "Specifying a region of influence") and uses a participation factor equal to one.

Weighting methods

You can modify the default weight distribution defined above. Various weighting methods are provided that monotonically decrease with radial distance from the reference node. For each case the default weight distribution that is based on the tributary surface area (or tributary edge length along a shell edge) is scaled by the weight factor w_i. If the weighting method is not specified, a uniform weighting method is used in which all weight factors are equal to 1.0.

Linearly decreasing weight distribution

A linearly decreasing weighting scheme

$$w_i = 1 - \frac{r_i}{r_0},$$

where w_i is the weight factor at coupling node i, r_i is the coupling node radial distance from the reference node, and r_0 is the distance to the furthest coupling node.

Input File Usage: *DISTRIBUTING, WEIGHTING METHOD=LINEAR

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Coupling**: **Coupling type**: **Distributing**: **Weighting method**: **Linear**

Quadratic polynomial weight distribution

A quadratic polynomial weight distribution defined by

$$w_i = 1 - \left(\frac{r_i}{r_0}\right)^2.$$

Input File Usage: *DISTRIBUTING, WEIGHTING METHOD=QUADRATIC

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Coupling**: **Coupling type**: **Distributing**: **Weighting method**: **Quadratic**

Monotonically decreasing weight distribution

A monotonically decreasing weight distribution according to the cubic polynomial

$$w_i = 1 - 3\left(\frac{r_i}{r_0}\right)^2 + 2\left(\frac{r_i}{r_0}\right)^3.$$

Input File Usage: *DISTRIBUTING, WEIGHTING METHOD=CUBIC

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Coupling**: **Coupling type**: **Distributing**: **Weighting method**: **Cubic**

Specifying a local coordinate system

The distributing coupling constraint can be specified with respect to a local coordinate system instead of the global coordinate system (see "Orientations," Section 2.2.5). Figure 33.3.2–2 illustrates the use of a local coordinate system to release the moment constraints between the reference node and the coupling nodes in the local 4- and 6-directions, providing a "universal-like" rotation connection. In this example a local rectangular coordinate system is defined that has its local y-axis coincident with the global z-axis. The moment constraint is specified in this local coordinate system.

Input File Usage: *COUPLING, ORIENTATION=*local*

For example, the following input is used to specify the distributing coupling constraint shown in Figure 33.3.2–2:

```
*ORIENTATION, SYSTEM=RECTANGULAR, NAME=COUPLEAXIS
0.0, 1.0, 0.0, 0.0, 0.0, 1.0
*COUPLING, REF NODE=500, SURFACE=Endcap,
ORIENTATION=COUPLEAXIS
*DISTRIBUTING
1, 3
5, 5
```

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Coupling**: **Edit**: select local coordinate system

Defining the surface coupling method

There are two methods available to couple the motion of the reference node to the average motion of the coupling nodes: the continuum coupling method and the structural coupling method. The continuum coupling method is used by default.

Continuum coupling method

The default continuum coupling method couples the translation and rotation of the reference node to the average translation of the coupling nodes. The constraint distributes the forces and moments at the reference node as a coupling nodes force distribution only. No moments are distributed at the coupling nodes. The force distribution is equivalent to the classic bolt pattern force distribution when the weight factors are interpreted as bolt cross-section areas. The constraint enforces a rigid beam connection between the attachment point and a point located at the weighted center of position of the coupling nodes. For further details, see "Distributing coupling elements," Section 3.9.8 of the Abaqus Theory Manual.

Input File Usage: *DISTRIBUTING , COUPLING=CONTINUUM

Abaqus/CAE Usage: Coupling the motion of the reference node to the average motion of the coupling nodes is not supported in Abaqus/CAE.

Structural coupling method

The structural coupling method couples the translation and rotation of the reference node to the translation and the rotation motion of the coupling nodes. The method is particularly suited for bending-like applications of shells when the coupling constraint spans small patches of nodes and the reference node is chosen to be on or very close to the constrained surface. The constraint distributes forces and moments at the reference node as a coupling node-force and moment distribution. For this coupling method to be active, all rotation degrees of freedom at all coupling nodes must be active (as would be the case when the constraint is applied to a shell surface) and the constraints must be specified in all degrees of freedom (default). In addition, for the constraint to be meaningful, the local (or global)

z-axis used in the constraint should be such that it is parallel to the average normal direction of the constrained surface.

With respect to translations, the constraint enforces a rigid beam connection between the reference node and a moving point that remains at all times in the vicinity of the constrained surface. The location of this moving point is determined by the approximate current curvature of the surface, the current location of the weighted center of position of the coupling nodes (see "Distributing coupling elements," Section 3.9.8 of the Abaqus Theory Manual), and the z-axis used in the constraint. This choice avoids unrealistic contact interactions if multiple pairs of distributed coupling constraints are used to fasten shell surfaces (see "Breakable bonds," Section 35.1.9, for more details).

With respect to rotations, the constraint is different along different local directions. Along the z-axis (twist direction), the constraint is identical to the one enforced via the continuum coupling method (see "Distributing coupling elements," Section 3.9.8 of the Abaqus Theory Manual). By contrast, the rotational constraint in the plane perpendicular to the z-axis relates the in-plane reference node rotations to the in-plane rotations of the coupling nodes in the immediate vicinity of the reference node. This choice provides a more realistic (compliant) response when the constrained surface is small and deforms primarily in a bending mode.

Input File Usage: *DISTRIBUTING, COUPLING=STRUCTURAL

Abaqus/CAE Usage: Coupling the motion of the reference node to the average motion of the coupling nodes is not supported in Abaqus/CAE.

Moment release and finite rotation

In geometrically nonlinear analysis steps the coordinate system of the degrees of freedom that define the moment release rotates with the reference node regardless of whether the global coordinate system or a local coordinate system is used.

Colinear coupling node arrangements

The distributing coupling constraint transmits moments at the reference node as a force distribution among the coupling nodes, even if these nodes have rotational degrees of freedom. Thus, when the coupling node arrangement is colinear, the constraint is not capable of transmitting all components of a moment at the reference node. Specifically, the moment component that is parallel to the colinear coupling node arrangement will not be transmitted. When this case arises, a warning message is issued that identifies the axis about which the element will not transmit a moment.

Limitations

- A distributing coupling constraint cannot be used with axisymmetric elements with asymmetric deformation. This element type is not compatible with the distributing coupling constraint.
- If a distributing coupling constraint is used with axisymmetric elements with twist, the constraint will not include the twist degree of freedom 5 in those elements. It will involve only the displacement degrees of freedom 1 and 2.

- A distributing coupling definition with a large number of coupling nodes produces a large wavefront in Abaqus/Standard. This may result in significant memory usage and a long solution time to solve the finite element equilibrium equations.
- A distributing coupling constraint cannot involve more than 46,000 degrees of freedom in Abaqus/Standard, which implies an upper limit of 23,000 nodes per constraint for two-dimensional and axisymmetric cases and an upper limit of 15,333 nodes per constraint for three-dimensional cases.

33.3.3 SHELL-TO-SOLID COUPLING

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Coupling constraints," Section 33.3.2
- "Surfaces: overview," Section 2.3.1
- *SHELL TO SOLID COUPLING
- "Defining shell-to-solid coupling constraints," Section 15.15.7 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Surface-based shell-to-solid coupling:

- allows for a transition from shell element modeling to solid element modeling;
- is most useful when local modeling should use a full three-dimensional analysis but other parts of the structure can be modeled as shells;
- uses a set of internally defined distributing coupling constraints to couple the motion of a "line" of nodes along the edge of a shell model to the motion of a set of nodes on a solid surface;
- automatically selects the coupling nodes located on a solid surface lying within a region of influence;
- can be used with three-dimensional stress/displacement shell and solid (continuum) elements;
- does not require any alignment between the solid and shell element meshes; and
- can be used in geometrically linear and nonlinear analysis.

Shell-to-solid coupling

Shell-to-solid coupling in Abaqus is a surface-based technique for coupling shell elements to solid elements. Figure 33.3.3–1 illustrates two examples taken from "Shell-to-solid submodeling and shell-to-solid coupling of a pipe joint," Section 1.1.10 of the Abaqus Example Problems Manual, and "The pinched cylinder problem," Section 2.3.2 of the Abaqus Benchmarks Manual. Shell-to-solid coupling is intended to be used for mesh refinement studies where local modeling requires a relatively fine through-the-thickness solid mesh coupled to the edge of a shell mesh, as shown in Figure 33.3.3–2. In such a case Abaqus will assemble constraints that couple the displacement and rotation of each shell node to the average displacement and rotation of the solid surface in the vicinity of the shell node.

As shown in Figure 33.3.3–2, the coupling occurs along a shell-to-solid interface defined by two user-specified surfaces: an edge-based shell surface and an element- or node-based solid surface (see "Surfaces: overview," Section 2.3.1). The shell surface (Figure 33.3.3–3) is referred to as the "shell

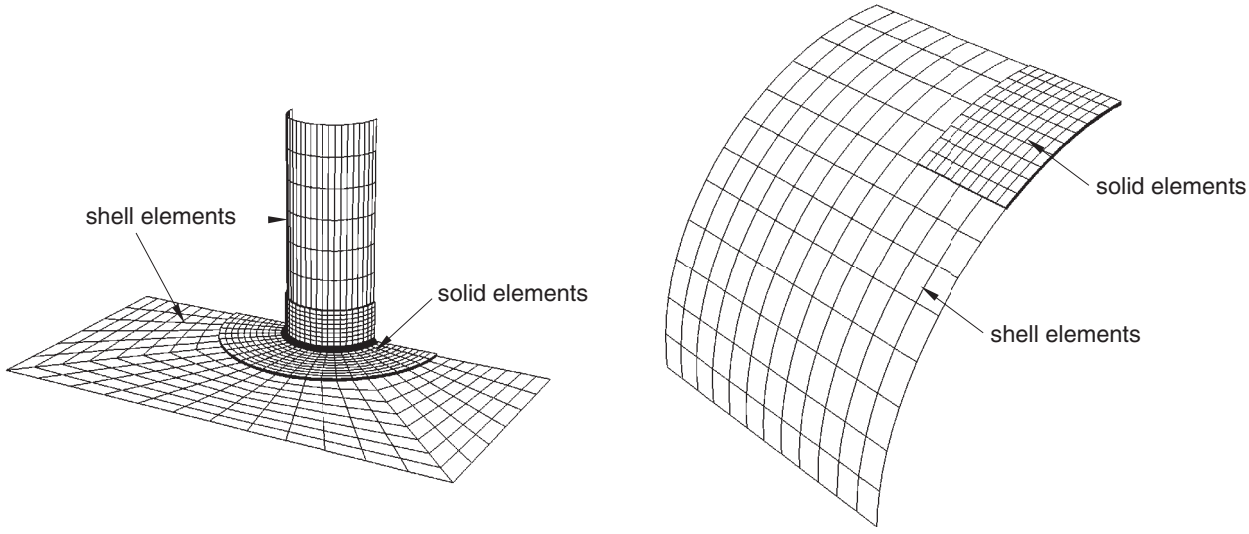


Figure 33.3.3–1 Typical examples of shell-to-solid coupling.

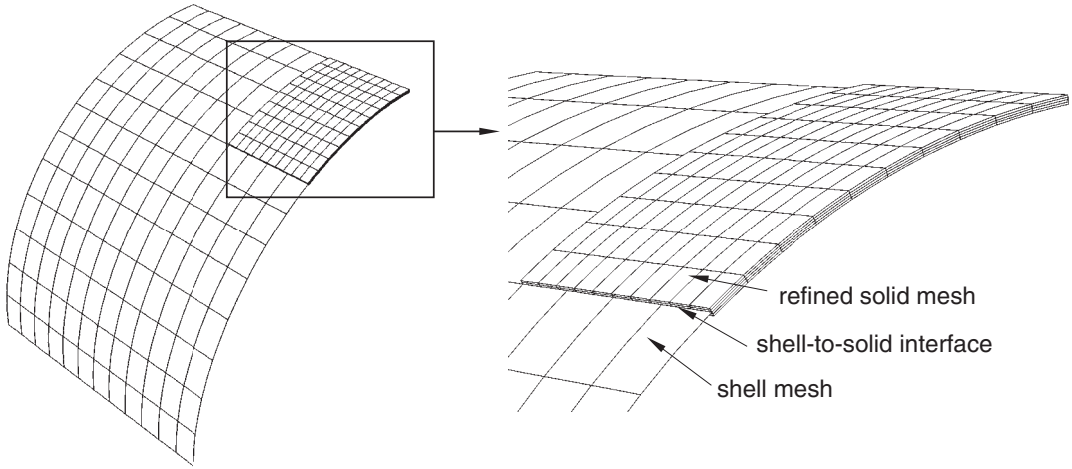


Figure 33.3.3–2 Shell-to-solid interface.

edge." The shell element edges that define the edge-based shell surface are referred to as "edge facets." The edge facets are either linear or parabolic segments depending if the underlying shell elements are linear or quadratic.

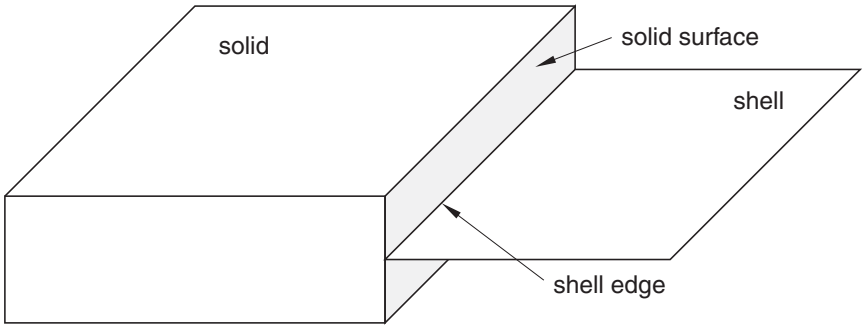


Figure 33.3.3–3 Shell and solid surfaces.

The shell-to-solid coupling is enforced by the automatic creation of an internal set of distributing coupling constraints (see "Coupling constraints," Section 33.3.2) between nodes on the shell edge and nodes on the solid surface. Abaqus uses default or user-defined distance and tolerance parameters (discussed below) to determine which nodes on the shell edge will be coupled to which nodes on the solid surface. For each shell node involved in the coupling, a distinct internal distributing coupling constraint is created with the shell node acting as the reference node and the associated solid nodes acting as the coupling nodes. Each internal constraint distributes the forces and moments acting at its shell node as forces acting on the related set of coupling surface nodes in a self-equilibrating manner. The resulting line of constraints enforces the shell-to-solid coupling.

Defining shell-to-solid coupling

Defining a shell-to-solid coupling constraint requires the specification of a constraint name, an edge-based shell surface, and an element- or node-based solid surface.

Input File Usage: *SHELL TO SOLID COUPLING, CONSTRAINT NAME=*name*
shell_surface, *solid_surface*

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Shell-to-solid coupling**

Abaqus automatically determines which nodes on the two surfaces participate in the coupling and creates appropriate internal distributed coupling constraints. You can also control which nodes on the two surfaces participate in the coupling by specifying a position tolerance and/or influence distance as described below.

The resulting coupling constraint definitions are printed to the data file when model definition data are requested (see "Controlling the amount of analysis input file processor information written to the data file" in "Output," Section 4.1.1). Abaqus will also create an internal node set that contains all the solid nodes included in the coupling; the node set can be visualized using the Visualization module of Abaqus/CAE. The name of the internal node set is the name assigned to the coupling constraint.

Controlling the shell nodes included in the coupling

A *position tolerance* determines the absolute distance from the solid surface within which all shell nodes to be included in the coupling must lie. Shell nodes that lie outside this tolerance are not coupled to the solid surface.

When using an element-based solid surface, the defined distance between a shell node and the solid surface is the projected distance measured along a line extending from the shell node to the closest point on the solid surface (which may be on the edge of the solid surface). The default position tolerance when using an element-based solid surface is 5% of the length of a typical facet on the shell edge.

For a node-based solid surface the defined distance of a shell node to the surface is the distance to the closest node on the solid surface. The default position tolerance when using a node-based solid surface is based on the average distance between nodes on the solid surface.

You can specify a nondefault position tolerance for element- or node-based solid surfaces..

Input File Usage: *SHELL TO SOLID COUPLING, POSITION TOLERANCE=*distance*

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Shell-to-solid coupling**: select the surfaces: choose **Specify distance** for the **Position Tolerance**

Controlling the solid nodes included in the coupling

A geometric tolerance, which is referred to as the *influence distance*, is defined for each edge facet. For a given node or element facet on the solid surface to be included in the coupling constraint, its perpendicular distance from at least one edge facet must be less than or equal to the influence distance defined for that edge facet. The default influence distance for an edge facet is half the thickness of the underlying shell element. The default automatically accounts for any offset or nodal thickness included with the shell element's cross-section definition. You can specify a nondefault influence distance.

Input File Usage: *SHELL TO SOLID COUPLING, INFLUENCE DISTANCE=*distance*

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Shell-to-solid coupling**: select the surfaces: choose **Specify value** for the **Influence Distance**

A user-defined influence distance is optional in all cases except when an edge facet involved in the coupling is associated with a general arbitrary elastic shell section definition in which you specified the general stiffness. In this case since the shell thickness is not defined directly, you must supply an influence distance.

Computation of the internal coupling constraints

This section outlines the basic procedure used by Abaqus to compute the internal shell-to-solid coupling constraints.

A single distinct internal distributing coupling constraint is created for each shell node that lies within the position tolerance from the solid surface. Internal coupling constraints are not created for shell nodes that lie outside this tolerance. The shell node acts as the reference node, and a set of nodes on the solid surface act as the coupling nodes. Abaqus finds the coupling nodes on the solid surface and

computes the weight factors for the internal constraints by considering each shell edge facet separately. The following procedure is carried out for each edge facet:

1. Abaqus finds all nodes on the solid element surface that lie within the region of influence (discussed below) of the current edge facet. These nodes are included in the coupling constraint.
2. Abaqus then computes a set of weight factors for the solid nodes. A weight factor is a measure of both the tributary area of the solid node contained within the region of influence and the relative position of the solid node with respect to each shell node. The tributary areas for node-based surfaces are the cross-sectional areas that you specified when you defined the surface. For element-based surfaces the tributary areas are calculated by Abaqus. The sum of all the weight factors in each coupling constraint is a measure of the total tributary area of the solid surface that is contained within the region of influence.
3. The above procedure is carried out for all the shell edge facets contained within the shell surface. If a shell node belongs to more than one edge facet, all the coupling nodes and weight factors are combined into a single distributing constraint definition. The resulting line of constraints along the shell edge enforces the shell-to-solid coupling.

There are two situations in which a shell node might satisfy the position tolerance but no coupling constraint is defined. If a shell node lies within the position tolerance but is not connected by an edge facet to at least one other shell node that also satisfies the tolerance, a coupling constraint is not created for this shell node. In this case it may be necessary to increase the position tolerance. Alternatively, if nonzero weight factors are not computed for at least two solid nodes associated with the shell node, a coupling constraint is not created for this shell node. The most likely cause for zero weight factors is that the influence distance is too small. In the case of a node-based surface, zero weights might also arise if the default cross-sectional area is used. For shell-to-solid coupling the default area is zero.

The region of influence for an edge facet

The region of influence of an edge facet is defined by a cylindrical volume whose centerline is the edge facet and whose radius is the edge facet's influence distance. The ends of the cylindrical volume are defined by two bounding planes whose normals are the shell tangents at the two ends of the edge facet (see Figure 33.3.3–4). In this example a region of influence is constructed for shell edge 2–3. For a node-based solid surface only the nodes that lie within or on the boundary of the region of influence are assigned to the current edge facet and included in the coupling definition. For an element-based solid surface each solid facet node is associated with part of the facet surface. If the part of the facet assigned to a given solid node falls within the region of influence, that node is included in the coupling definition.

Using the normal on an element-based solid surface to restrict solid nodes that are used in the coupling

In the case of an element-based solid surface Abaqus will compare the normal of each solid facet within the region of influence to the normal of the solid surface closest to the centerline of the cylindrical volume (see Figure 33.3.3–4). In general, if the normal of a surface facet is not within 20° of the normal at the centerline, the nodes on the solid surface facet are not included in the coupling definition. For the case illustrated in Figure 33.3.3–4 this check would prevent nodes on the top and bottom surface of the solid

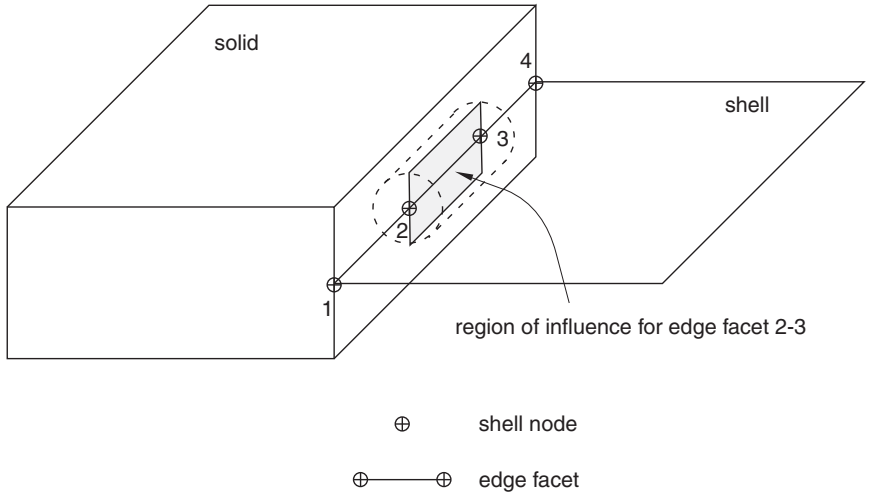


Figure 33.3.3–4 Regions of influence for an edge facet.

mesh from being coupled to the shell nodes even if the influence distance was arbitrarily large and the solid surface definition included all sides of the solid geometry. This check is not used if the centerline is on or near a feature edge of the solid mesh where the normal is not well defined (see the discussion about shell offsets below).

Comments, restrictions, and modeling recommendations for shell-to-solid coupling

- The shell-to-solid coupling formulation assumes that the interface surface between the shell and solid elements is normal to the shell. Therefore, while the solid surface can be curved in a direction tangent to the shell edge, it should be straight in the direction along the shell normals. This is an assumption on the geometry of the surfaces, not on the mesh. It is not necessary for the nodes on the solid surface to line up with each other or to line up with the shell nodes.
- The shell-to-solid coupling capability is designed for analyses where the solid mesh is fine with respect to the shell thickness. It is recommended that at least two solid elements be included through the thickness at a shell-to-solid interface. Along the shell-to-solid interface the length of a shell edge facet should in general be of the same order as the characteristic surface dimension of a solid element facet.
- An assumption used in the design of the shell-to-solid coupling algorithms is that the weight factors are based upon accurate nodal tributary areas, such as those automatically computed by Abaqus when an element-based surface is used. Therefore, it is generally recommended that an element-based solid surface be used instead of a node-based solid surface. However, in cases where the shell and solid meshes align with each other, it is sometimes advantageous to use a node-based solid surface especially when a homogenous solution is expected.

- Figure 33.3.3–5 illustrates some recommended modeling practices for shell-to-solid coupling. If the shell reference surface is not offset, the shell edge should be centrally located with respect to the thickness direction of the solid (Figure 33.3.3–5(a)). The solid surface should include only the portion needed for the coupling (the shaded region shown in Figure 33.3.3–5(a)).

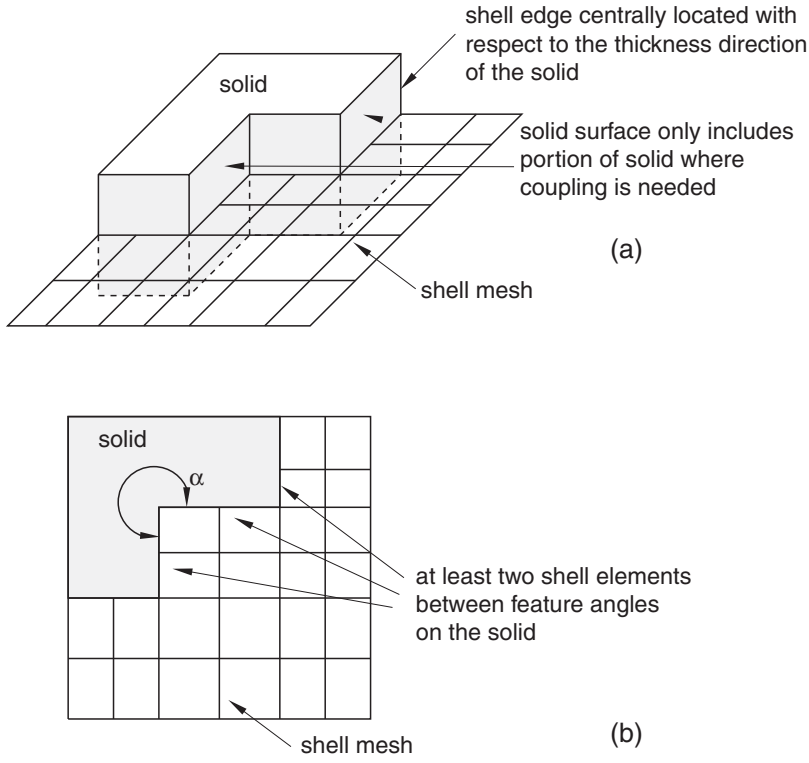


Figure 33.3.3–5 Modeling recommendations for the shell-to-solid interface.

- The shell-to-solid interface can be defined around geometric feature angles (corners), α (Figure 33.3.3–5(b)). However, it is recommended that the feature angles satisfy $60° < \alpha < 300°$. In addition, as illustrated in Figure 33.3.3–5(b), at least two shell element edges should be included between each feature angle.
- If an offset is defined for the shell section and the reference shell edge is placed at or near a feature edge on the solid surface (Figure 33.3.3–6), the solid surface should include only the side of the solid that you want to be included in the coupling definition.

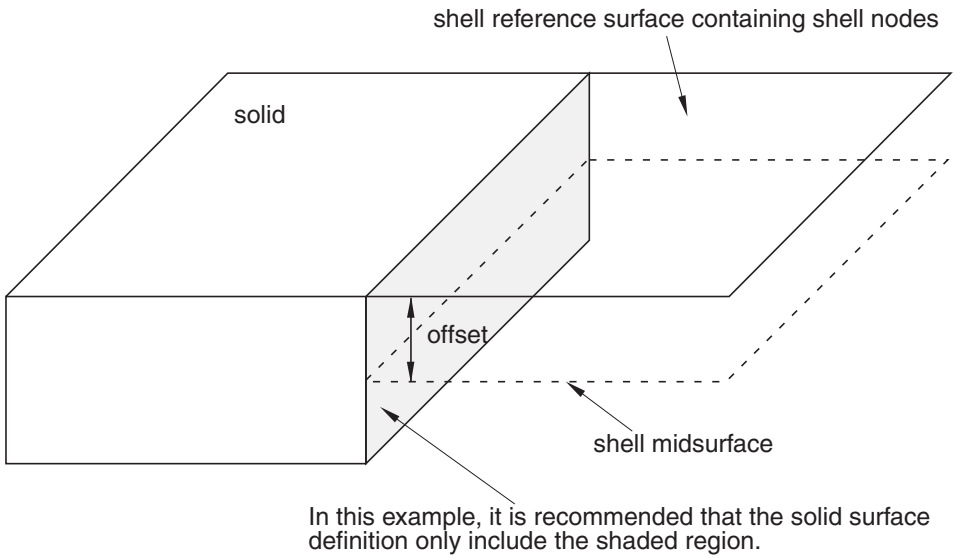


Figure 33.3.3–6 Modeling recommendations for the shell-to-solid interface with a shell offset.

For example, if the top of the solid in Figure 33.3.3–6 is included in the surface definition, Abaqus includes nodes on the top of the surface in the coupling constraint, which is not what you intended. You intended only that the shell be coupled to the shaded region of the solid in Figure 33.3.3–6. Therefore, the solid surface definition should include only this region.

- Care must be taken in interpreting the local stress and strain fields in the immediate vicinity of the shell-to-solid interface. This is especially true if the shell-to-solid interface includes corners or edges. The interface should be placed at least a distance more than the shell thickness away from the region in the solid mesh where the stress and strain fields are of interest.
- The shell-to-solid interface should be located in a region of the model where shell theory is a valid modeling approximation.
- Corners or kinks may exist in models made of shell elements. At such corners or kinks the shell elements only approximate the distribution of the material away from the midsurface of the shell. While the global moments and forces between the shell and solid models are transferred correctly, the local stress and displacement fields in the region of the shell-to-solid interface may be inaccurate.
- Only displacement degrees of freedom in the solid elements and displacement and rotation degrees of freedom in the shell elements are coupled in shell-to-solid coupling. Shell-to-solid coupling does not couple other degrees of freedom such as temperature, pressure, etc.
- Shell-to-solid coupling can be used to couple three-dimensional shells to all three-dimensional continuum elements except cylindrical elements (“Cylindrical solid element library,” Section 27.1.5).

33.3.4 MESH-INDEPENDENT FASTENERS

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Surfaces: overview," Section 2.3.1
- "Coupling constraints," Section 33.3.2
- "Connector elements," Section 30.1.2
- *FASTENER
- *FASTENER PROPERTY
- "About fasteners," Section 29.1 of the Abaqus/CAE User's Manual

Overview

The mesh-independent fastener capability:

- is a convenient method to define a point-to-point connection between two or more surfaces such as a spot weld or rivet connection;
- uses spatial coordinates of fastener locations to define point-to-point connections independent of underlying meshes;
- combines either connector elements or BEAM MPCs with distributing coupling constraints to provide a connection that can be located anywhere between two or more surfaces regardless of the mesh refinement or location of nodes on each surface;
- can be used to connect both deformable and rigid element-based surfaces;
- can model either rigid, elastic, or inelastic connections with failure by using the generality of connector behavior definitions; and
- is available only in three dimensions.

Introduction

Many applications require modeling of point-to-point connections between parts. These connections may be in the form of spot welds, rivets, screws, bolts, or other types of fastening mechanisms. There may be hundreds or even thousands of these connections in a large system model such as an automobile or airframe.

The fastener can be located anywhere between the parts that are to be connected regardless of the mesh. In other words, the location of the fastener can be independent of the location of the nodes on the surfaces to be connected. Instead, the attachment to each of the parts being connected is distributed to several nodes in the surfaces to be connected in the neighborhood of the fastening points. Figure 33.3.4–1 shows a typical one-layer and two-layer fastener configuration. Each layer connects two fastening points using either a connector element or a BEAM MPC. Each fastening point is connected to the surface using

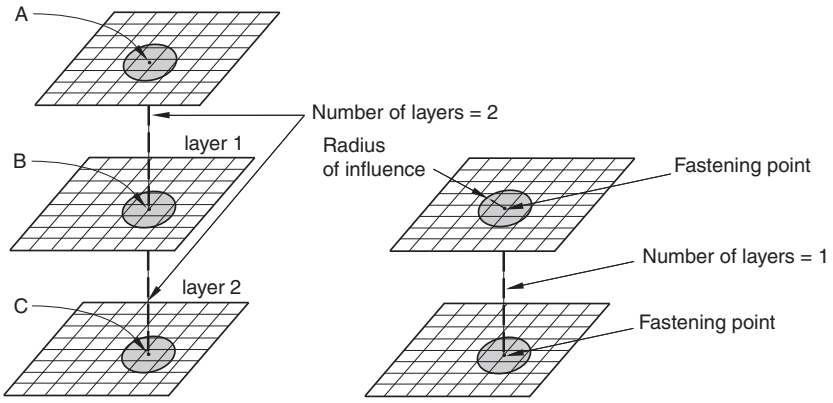


Figure 33.3.4–1 Typical one-layer and two-layer fastener configuration.

a distributing coupling constraint that couples the displacement and rotation of each fastening point to the average displacement and rotation of the nearby nodes.

The mesh-independent fastener capability in Abaqus is designed to model these connections in a convenient manner. The fastener automatically:

- determines the locations of nodes and orientations of connector elements or BEAM MPCs between two or more surfaces;
- generates distributing coupling constraints to attach the connector elements or BEAM MPCs to each surface in a mesh-independent manner; and
- calculates weights for the distributing coupling constraints that complete the mesh-independent connection.

For an example of the use of mesh-independent fasteners, see "Buckling of a column with spot welds," Section 1.2.3 of the Abaqus Example Problems Manual. Mesh-independent fasteners are referred to as point-based fasteners by Abaqus/CAE. For more information, see "About fasteners," Section 29.1 of the Abaqus/CAE User's Manual. It is also possible to assemble fasteners in Abaqus/CAE using connector elements, coupling constraints, etc. For further details, see "About assembled fasteners," Section 29.1.3 of the Abaqus/CAE User's Manual.

Fastener interactions

Fasteners are defined in groups called interactions, which are assigned names. Each interaction defines one or more fasteners. The number of individual fasteners is equal to the number of positioning points used to locate the fasteners. Fastening points on each surface are found by considering the position of the positioning point as discussed in subsequent sections.

Fasteners can be defined using connector elements or BEAM MPCs. BEAM MPCs allow modeling of perfectly rigid connectors between components; while connector elements allow you to model much more complex behavior, such as deformable connectors that include the effects of elasticity, damage, plasticity, and friction.

Input File Usage: *FASTENER, INTERACTION NAME=*name*

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Name**: *name*, **Type**: **Point-based**

Defining fasteners using BEAM MPCs

For modeling perfectly rigid connections you need not define fasteners using connector elements. Instead, Abaqus can internally generate BEAM MPCs connecting the fastening points of the fasteners. In this approach you assign a reference node set containing a list of user-defined nodes to the fastener interaction. The nodes in this reference node set will be used as positioning points to locate the fasteners. If single-layer fasteners are to be modeled, Abaqus generates single BEAM MPCs with each node in the reference node set becoming the first node of the BEAM MPC. The second node of each BEAM MPC will be generated internally by Abaqus. If multi-layer fasteners are to be defined, Abaqus generates linked sets of BEAM MPCs with each node in the reference node set becoming the first node of the first BEAM MPC in each linked set. The subsequent nodes in each linked set will be generated internally by Abaqus. For multi-layer fasteners each linked set contains as many BEAM MPCs as the number of layers in the fastener.

Input File Usage: Use the following options:

*FASTENER, INTERACTION NAME=*name*,
REFERENCE NODE SET=*node set label*
*NSET, NSET=*node set label*

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: select positioning points: **Property**: **Section**: **Rigid MPC**

Defining fasteners using connector elements

Using connector elements as the basis for a point-to-point connection allows for very complex behavior to be modeled with fasteners. Like other uses of connector elements, the connection can be fully rigid or may allow for unconstrained relative motion in local connector components. In addition, deformable behavior can be specified using a connector behavior definition that can include the effects of elasticity, damping, plasticity, damage, and friction. There are two methods to define fasteners that use connector elements to model the behavior between fastening points. For both methods the fastener interaction refers to an element set containing the connector elements. You must specify a connector section definition that refers to this element set. You should be careful when specifying the connector orientation (if needed) as discussed below in "Defining the fastener orientation."

Defining the connector elements directly

The most controlled approach to specifying fasteners using connector elements is to define the connector elements explicitly and associate them with an element set. The fastener interaction refers to the element set. Each fastener in the fastener interaction corresponds to one or more connector elements depending on the number of layers of the fastener (see Figure 33.3.4–2). A single connector element is associated with each layer, and the two nodes of the connector element correspond to the fastening points of the two adjacent surfaces. When specifying a multi-layer fastener, the connector elements for each layer should share nodes with the connector elements of adjacent layers.

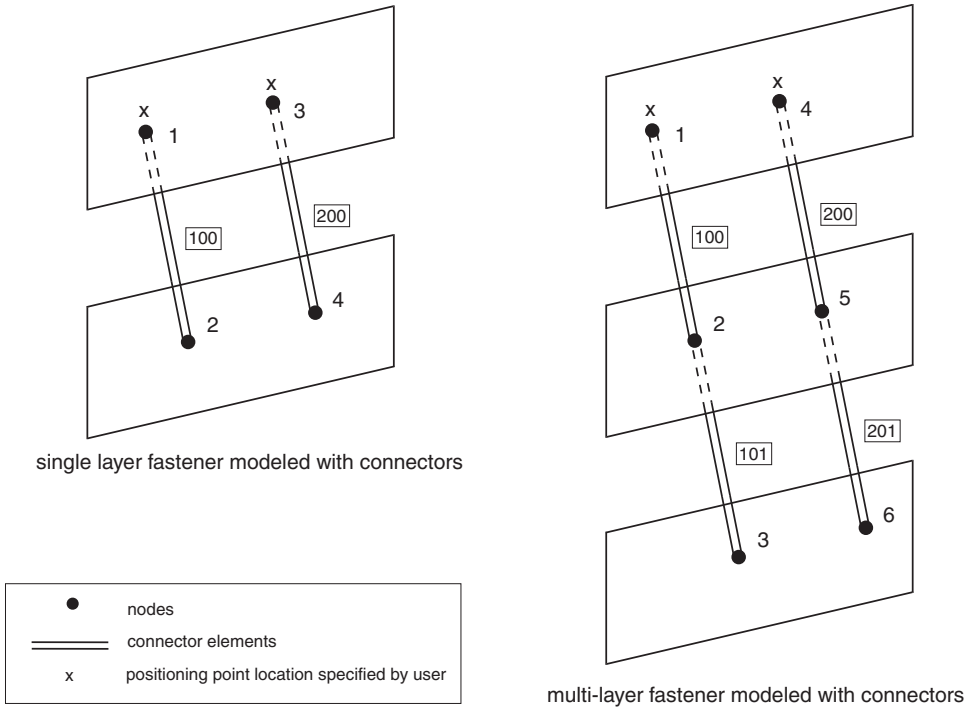


Figure 33.3.4–2 Single- and multi-layer fasteners modeled with connector elements.

For a single-layer fastener the positioning point used to locate the fastener and its fastening points is taken as the nodal coordinates of the first node of the connector element. For a multi-layer fastener the positioning point is taken as the first node of the first connector in a linked set of connectors with as many members as layers. Examples of defining a single-layer and multi-layer fastener are included at the end of this section.

Input File Usage: Use the following options:

∗FASTENER, INTERACTION NAME=*name*, ELSET=*element set label*
blank line
∗ELEMENT, TYPE=CONN3D2, ELSET=*element set label*
∗CONNECTOR SECTION, ELSET=*element set label*

Abaqus/CAE Usage: For point-based fasteners in Abaqus/CAE, you cannot define the connector elements directly; the connector elements are generated by Abaqus.

Connector elements generated by Abaqus

In this approach you do not need to explicitly define the connector elements that connect the fastening points of the fastener. The fastener interaction refers to an empty element set. You must specify a

connector section definition that refers to this element set. In addition, you assign a reference node set containing a list of user-defined nodes to the fastener interaction. The nodes in this reference node set are used as positioning points to locate the fasteners.

If single-layer fasteners are to be modeled, Abaqus generates single connector elements with each node in the reference node set becoming the first node of a connector element. The second node of each connector element will be generated internally by Abaqus. If multi-layer fasteners are to be defined, Abaqus generates linked sets of connector elements with each node in the reference node set becoming the first node of the first connector element in each linked set. The subsequent nodes in each linked set will be generated internally by Abaqus. For multi-layer fasteners each linked set contains as many connector elements as the number of layers in the fastener. The connector elements are given internally generated element numbers and assigned to the named user-specified element set. You can use this element set to request output for these connector elements. However, this element set should not be included in another element set definition.

Input File Usage: Use the following options:

∗FASTENER, INTERACTION NAME=*name*, ELSET=*element set label*, REFERENCE NODE SET=*node set label*
blank line
∗NSET, NSET=*node set label*
∗CONNECTOR SECTION, ELSET=*element set label*

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: select positioning points: **Property**: **Section**: **Connector section**: select connector section

Example: using connector elements to define single-layer fasteners directly

To define a single-layer fastener directly using connector elements:

- Define two connector elements with user element numbers 100 and 200 and user-defined node numbers 1, 2 and 3, 4, respectively, and include them in an element set. Nodes 1 and 3 act as the positioning points for the two fasteners (see Figure 33.3.4–2).
- Refer to the element set in the fastener interaction and connector section definitions.
- Assign section properties to the fasteners. Suppose in this example that relative displacements between the fastening points are to be allowed. Therefore, the fasteners must be assigned a section that has available components of motion; for example, a CARTESIAN section can be used.
- The relative displacement between the fastening points gives rise to elastic deformations. Hence, the material between the fasteners is modeled as linear elastic with a spring stiffness of 10000 using connector elasticity.

The following input can be used:

```
*FASTENER, INTERACTION NAME=fastinter, ELSET=fastconn, PROPERTY=fastprop
blank line
surface1, surface2
*ELEMENT, TYPE=CONN3D2, ELSET=fastconn
```

```
100, 1, 2
200, 3, 4
*CONNECTOR SECTION, ELSET=fastconn, BEHAVIOR=behav
CARTESIAN,
*CONNECTOR BEHAVIOR, NAME=behav
*CONNECTOR ELASTICITY, COMPONENT=1
10000,
*CONNECTOR ELASTICITY, COMPONENT=2
10000,
*CONNECTOR ELASTICITY, COMPONENT=3
10000,
```

Example: using connector elements to define multi-layer fasteners directly

To define a multi-layer fastener directly using connector elements:

- Define two linked sets of connector elements with each linked set containing exactly two connectors. The first linked set comprises element numbers 100 and 101, with node numbers 1, 2 and 2, 3, respectively. The second linked set comprises element numbers 200 and 201, with node numbers 4, 5 and 5, 6, respectively. Include the connector elements in an element set. Nodes 1 and 4 act as the positioning points for the two fasteners (see Figure 33.3.4–2).
- Refer to the element set in the fastener interaction and connector section definitions
- Assign section properties to the fasteners. Suppose in this example that rigid beam-type behavior between the fastening points is to be modeled; in that case the fasteners must be assigned a BEAM section.

The following input can be used:

```
*FASTENER, INTERACTION NAME=fastinter, ELSET=fastconn, PROPERTY=fastprop
blank line
surface1, surface2, surface3
*ELEMENT, TYPE=CONN3D2, ELSET=fastconn
100, 1, 2
101, 2, 3
200, 4, 5
201, 5, 6
*CONNECTOR SECTION, ELSET=fastconn
BEAM,
```

Specifying the positioning points, projection method, and fastening points

Each interaction defines one or more fasteners. The number of individual fasteners is equal to the number of positioning points used to locate the fasteners. Positioning points are nodes defined at the fastener locations and assigned as a reference node set to the interaction.

In general, a positioning point should be located as close to the surfaces being connected as possible. The reference node specifying the positioning point can be one of the nodes on the connected surfaces

or can be defined separately. Abaqus determines the actual points where the fastener layers attach to the surfaces that are being connected by first projecting the positioning point onto the closest surface. Abaqus offers the following projection methods to find fastening points on the specified surfaces to form fasteners:

- Face-to-face
- Face-to-edge
- Edge-to-face
- Edge-to-edge

The choice of method depends on how the surfaces are oriented relative to each other.

Fastening surfaces that are nearly parallel to each other

Most commonly the surfaces to be fastened together are nearly parallel to each other; in which case the fastening points are located on element facets away from the periphery of the surfaces. The face-to-face projection method is most appropriate for such situations. It is also the default projection method.

In the face-to-face projection method, Abaqus projects each positioning point onto the closest surface along a directed line segment normal to the surface. Alternatively, you can specify the projection direction. Specifying the direction may be useful when two-dimensional drawings are used to identify the positioning point locations and those locations are known precisely in two dimensions but not in a third. For this case the direction specified is typically the normal to the plane of the drawing.

Once the fastening point on the closest surface has been identified, Abaqus determines the points on the other surface or surfaces to be connected by projecting the first fastening point onto the other surfaces along the fastener normal direction, which is typically normal to the closest surface. Figure 33.3.4–3 shows the two ways of locating the projection points.

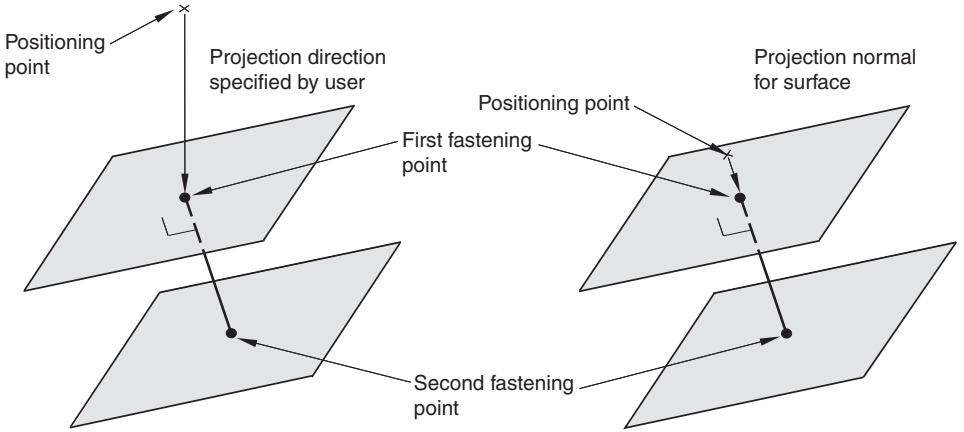


Figure 33.3.4–3 Directed and normal projection to locate the fastening points for the face-to-face projection method.

The location of the positioning point (a node in the reference node set) might not coincide with the locations of the fastening points found by Abaqus. Hence, the coordinates of the node at the positioning point may change from their user-prescribed values when the node is shifted to a fastening point. If the node at the positioning point is part of the connectivity of a user-defined element, this can cause the element whose connectivity includes that node to undergo unacceptable initial distortions. In such situations it is recommended that you define the node at the positioning point separately. In general, you should not specify this node to be one of the nodes of the connected surfaces.

Input File Usage: Use the following option to allow Abaqus to define the projection direction:

∗FASTENER, REFERENCE NODE SET=*node set label*, ATTACHMENT METHOD=FACETOFACE (default)
blank line

Use the following option to define the projection direction directly:

∗FASTENER, REFERENCE NODE SET=*node set label*, ATTACHMENT METHOD=FACETOFACE (default)
x-component, y-component, z-component

Abaqus/CAE Usage: Use the following input to allow Abaqus to define the projection direction:

Interaction module: **Special→Fasteners→Create**: **Point-based**: select positioning points: **Domain** tabbed page: **Direction vector**: **Default**, **Criteria** tabbed page: **Attachment method**: **Face-to-Face**

Use the following input to define the projection direction directly:

Interaction module: **Special→Fasteners→Create**: **Point-based**: select positioning points: **Domain** tabbed page: **Direction vector**: **Specify**, **Criteria** tabbed page: **Attachment method**: **Face-to-Face**

Fastening surfaces that are normal to each other

When you need to fasten surfaces that are oriented normal to each other; i.e., forming a T-intersection, the face-to-edge or the edge-to-face projection methods are appropriate choices. Figure 33.3.4–4 shows attachments for the face-to-edge and edge-to-face projection methods.

Creating the first fastening point on a face

In the face-to-edge projection method Abaqus projects the positioning point onto the closest surface along a directed line segment normal to the surface. The subsequent fastening points are found by searching for the closest points on the remaining specified surfaces. The closest fastening point may fall on the edge or a corner of a surface.

Input File Usage: ∗FASTENER, REFERENCE NODE SET=*node set label*, ATTACHMENT METHOD=FACETOEDGE
blank line

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: select positioning points: **Criteria**: **Attachment method**: **Face-to-Edge**

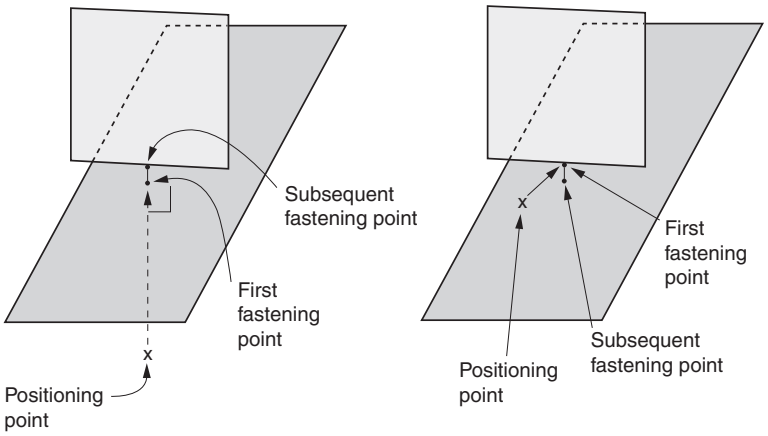


Figure 33.3.4–4 Face-to-edge and edge-to-face projection methods to locate fastening points for surfaces that form T-intersections.

Creating the first fastening point on an edge

In the edge-to-face projection method, the first fastening point is found by searching for the closest point on the specified surface or surfaces. The closest point may be on the edge or corner of the surface. For subsequent fastening points Abaqus projects the previous fastening point along a directed line segment normal to the surface.

Input File Usage: *FASTENER, REFERENCE NODE SET=*node set label*,
ATTACHMENT METHOD=EDGETOFACE
blank line

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: select positioning points: **Criteria**: **Attachment method**: **Edge-to-Face**

Fastening abutting surfaces

When it is desired to form fasteners between surfaces that are butting against each other, the edge-to-edge projection method is appropriate. In this method the first as well as the subsequent fastening points are located by searching for the closest point on the specified surface or surfaces. The fastening points in this method may be located on the edge of a surface. Figure 33.3.4–5 shows attachments for the edge-to-edge projection method.

Input File Usage: *FASTENER, REFERENCE NODE SET=*node set label*,
ATTACHMENT METHOD=EDGETOEDGE
blank line

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: select positioning points: **Criteria**: **Attachment method**: **Edge-to-Edge**

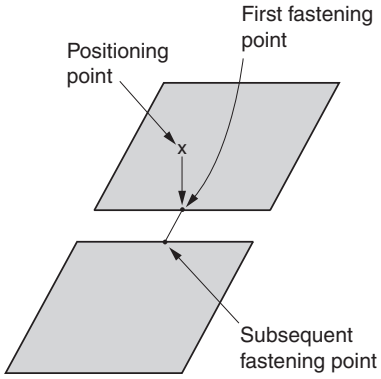


Figure 33.3.4–5 Edge-to-edge projection method to locate fastening points for abutting surfaces.

Specifying the surfaces to be fastened

Once the positioning points have been specified, the surfaces to be fastened can be specified using two different approaches. In the first approach you directly specify the surfaces that are to be connected with a fastener. In the second approach you specify a search zone, and Abaqus automatically identifies the surfaces that are to be connected. However, in the second approach Abaqus does not distinguish between coincident facets. Hence, if coincident facets are to be fastened, you should specify distinct surfaces containing each of the coincident facets and use the first approach. Only element-based surfaces defined on faces can be fastened together (see "Element-based surface definition," Section 2.3.2, and "Operating on surfaces," Section 2.3.6).

Forming fasteners on user-specified surfaces

If you specify multiple surfaces as part of the interaction definition, the surfaces to be fastened are restricted to these surfaces. The number of layers of each fastener is one less than the number of surfaces specified. One fastening point is found on each surface.

Input File Usage: *FASTENER
first data line
surface1, surface2, surface3, etc.

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: **Domain**: **Approach**: **Fasten specified surfaces by proximity**, select surfaces

When you select multiple surfaces for a single surface region, Abaqus/CAE combines the multiple surfaces using the single-surface search method, as described in "Forming fasteners on surfaces inside a user-specified search zone" below.

Controlling connectivity of fasteners on user-specified surfaces

By default, the connectivity of the fastening points is determined by their relative position along the fastener projection direction. For example, the default connectivity for the two-layer example shown in Figure 33.3.4–1 connects fastening point A to point B (layer 1) and point B to point C (layer 2).

You can control the connectivity of the fastening points when the fasteners are formed on user-specified surfaces. You can specify that the connectivity of the fastening points be defined by the order in which you specified their associated surfaces.

Input File Usage: *FASTENER, UNSORTED
first data line
surface1, surface2, surface3, etc.

If user-specified surfaces are not included on the data lines, the UNSORTED parameter is ignored.

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: **Domain**: **Approach**: **Fasten in specified order**, select surfaces

Forming fasteners on surfaces inside a user-specified search zone

If you do not specify any surfaces as part of the interaction definition, Abaqus searches for fastening points on all element facets that fall within a sphere of user-specified radius R with its center at the positioning point. If you do not specify the search radius, Abaqus computes a default search radius based on five times the facet thickness (for shell element facets) or the characteristic element length (for other element types) in the vicinity of each positioning point.

To refine the search, you can specify a single surface definition that will limit the facet search to element facets belonging to that surface. In this case you must define a collective surface that includes at least each connected surface. A combined surface can also be used (see "Operating on surfaces," Section 2.3.6, for a discussion on combining surfaces).

To refine the search further, you can specify a positive integer value, N, for the number of layers of each fastener. Abaqus searches for the $N + 1$ fastening points closest to the positioning point. If BEAM MPCs are used to model the fastener, a warning message is issued if the requisite number of fastening points is not found. However, if connector elements are used to model the fastener and the requisite number of fastening points is not found, Abaqus issues an error message. Thus, when specifying the number of layers, you should ensure that the search radius has been specified such that $N + 1$ fastening points can be found.

If multiple surfaces are listed as part of the fastener definition, the number of layers for each fastener is ignored. If a user-specified search radius is used for the multiple surface case, Abaqus searches for fastening points on all facets belonging to each of the listed surfaces that fall within a sphere of user-specified radius R with its center at the positioning point. Facets of the listed multiple surfaces that lie outside this sphere are not included in the search. A maximum of 15 layers can be specified for a particular fastener definition.

Input File Usage: *FASTENER, SEARCH RADIUS=*R*, NUMBER OF LAYERS=*N*
first data line

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: **Criteria**: **Search radius**: **Specify**: *R*, **Maximum layers for projection**: **Specify**: *N*

Defining the radius of influence

Each fastening point is associated with a group of nodes on the surface in the immediate neighborhood of the fastening point called a region of influence. The motion of the fastening point is then coupled in a weighted sense to the motion of the nodes in this region by a distributed coupling constraint. Several weighting options are available and are discussed in the next section.

To define the region of influence, Abaqus computes an internal radius of influence based on the geometric properties of the fastener, the characteristic length of the connected facets, and the type of weighting function used. The default radius of influence is always chosen to be the largest of the internally computed radius of influence, the physical fastener radius, and the distance of the projection point to the closest node. You can also specify the desired radius of influence. However, Abaqus overrides a user-specified radius of influence that is smaller than the computed default radius of influence. In any case each region of influence will contain a minimum of three nodes.

Input File Usage: ∗FASTENER, RADIUS OF INFLUENCE=*distance*
blank line

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: **Adjust**: **Influence radius**: **Specify**: *distance*

Defining the weighting method

The weighting methods available for the distributed coupling constraints created for a fastener interaction are the same as those available for the surface-based coupling constraints in Abaqus (see "Coupling constraints," Section 33.3.2). Besides an area-based uniform weighting scheme, various weighting methods are provided that monotonically decrease with radial distance from the fastening point: linear, quadratic, and cubic polynomial weight distributions. By default, Abaqus uses the uniform weighting method. You can modify the default weighting distribution.

The default radius of influence calculated by Abaqus is larger for higher-order weighting methods since the resulting weights for nodes away from the fastening point contribute comparatively little to the motion of the fastening point. Hence, to ensure that there is a sufficient "smearing" effect, it becomes necessary to increase the number of nodes in the region of influence by increasing the size of the default radius of influence. In comparison, for a uniform weighting scheme, surface nodes away from the fastening point contribute significantly to the motion of the fastening point. For this case the default radius of influence chosen can be comparatively small, since even with a small number of nodes in the region of influence, the smearing effect is sufficiently strong.

Input File Usage: Use the following option to specify a uniform weight distribution:

∗FASTENER, WEIGHTING METHOD=UNIFORM
blank line

Use the following option to specify a linear weight distribution:

∗FASTENER, WEIGHTING METHOD=LINEAR
blank line

Use the following option to specify a quadratic polynomial weight distribution:

*FASTENER, WEIGHTING METHOD=QUADRATIC
blank line

Use the following option to specify a cubic polynomial weight distribution:

*FASTENER, WEIGHTING METHOD=CUBIC
blank line

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: **Formulation**: **Weighting method**: **Uniform**, **Linear**, **Quadratic**, or **Cubic**

Defining the fastener orientation

Each fastener is formulated in a local coordinate system that rotates with the motion of the fastener. By default, Abaqus defines the local system by projecting the global coordinate system onto the surfaces that are being fastened according to the usual convention for surfaces in space (see "Conventions," Section 1.2.2). Local directions specified in this manner are such that the local z-axis for each fastener is normal to the surface that is closest to the reference node for the fastener.

You can override the default local system by specifying a local coordinate system for the fastener interaction. Generally, the user-defined orientation should be such that the local z-axis of the orientation is approximately normal to the surfaces that are being connected and the local x- and y-axes are approximately tangent to the surfaces that are being connected. By default, Abaqus adjusts the user-defined orientation such that the local z-axis for each fastener is normal to the surface that is closest to the reference node for the fastener. In cases where you wish to define the local directions precisely, you can specify that Abaqus should not adjust them.

Fasteners support only rectangular, cylindrical, and spherical orientation definitions. Additional rotations defined as part of the orientation definition are ignored.

In geometrically nonlinear analysis steps the local directions rotate with the motion of the fastener reference node.

Local coordinate system when connector elements are used

If a connector element is used to model a fastener, the local coordinate system defined on the connector section, $\mathbf{T}_{connector}$, operates on the local coordinate system for the fastener, $\mathbf{T}_{fastener}$, to determine the final local coordinate system of the connector element, $\mathbf{T}_{connectorfinal}$. In other words,

$$\mathbf{T}_{connectorfinal} = \mathbf{T}_{connector} * \mathbf{T}_{fastener}.$$

In the above equations $\mathbf{T}_{connector}$ and $\mathbf{T}_{fastener}$ are assumed to be orthogonal rotation matrices with the local 1-, 2-, and 3-directions being the first, second, and third rows, respectively. The local coordinate system for a connector element modeling a fastener should be specified with respect to the local coordinate system of the fastener. The orientation displayed in the Visualization module of Abaqus/CAE (Abaqus/Viewer) is $\mathbf{T}_{connectorfinal}$ at all fastener locations if connector field output is requested. If no field output is requested or if you specify not to have the orientations written to the database, only the $\mathbf{T}_{connector}$ is displayed.

For example, suppose you use a HINGE connector and want the released rotational degree of freedom, which is in the connector's local 1-direction, to be normal to the surfaces that are being fastenened. If the default local coordinate system is used for the fastener (local 3-direction normal to the surface), the local 1-direction for the connector should be set to (0., 0., 1.); i.e., the local 3-direction of the fastener. When compounded with the local coordinate system for the fastener, the local 1-direction for the connector will be normal to the surface. See "Mesh-independent spot welds," Section 5.1.16 of the Abaqus Verification Manual, for an example of a compounded orientation.

Input File Usage: *FASTENER, ORIENTATION=*orientation name*,
ADJUST ORIENTATION=NO
blank line

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: **Adjust**: **Fastener CSYS**: **Edit**: select local coordinate system, toggle off **Adjust CSYS to make local Z-axis normal to closest surface**

Clarifications regarding the computation of $\mathbf{T}_{fastener}$

A few clarifications regarding the default definition of $\mathbf{T}_{fastener}$ are necessary for a precise understanding of the behavior when connector elements are used to model fasteners. The positioning point is always projected on the closest surface to be fastened. Therefore, the choice of coordinates of the reference node relative to the stack of surfaces to be fastened determines which surface is used to compute the local directions. Typically this choice does not matter much in realistic applications because the surfaces to be fastened are more or less parallel to each other in the fastener area.

The projection of the reference node on the closest surface generates a fastening point for the connector element. The local z-axis for each fastener ($\mathbf{T}_{fastener}$) is normal to the surface at this fastening point. The fastening point generated on the closest surface is by default the first fastening point and, therefore, the first connector node. The precise direction into which the local z-axis is pointing is chosen such that the dot product with the unit vector pointing from the first node of the connector to the second node of the connector is positive. As explained above, you can control the connectivity of the fastening points in the connectors by specifying unsorted surfaces. Therefore, you can control the precise direction the local z-axis is pointing along the surface normal by either selecting appropriate coordinates for the reference node and/or by using unsorted surfaces.

The two tangential directions in $\mathbf{T}_{fastener}$ are computed by default according to the usual convention for surfaces in space (see "Conventions," Section 1.2.2). The global X-axis is projected onto the closest surface at the location of the fastening point to determine the local x-axis in $\mathbf{T}_{fastener}$. If the global X-axis is within 0.1 degrees of being normal to the surface, the local x-axis in $\mathbf{T}_{fastener}$ is the projection of the global Z-axis on the closest surface. The local y-axis in $\mathbf{T}_{fastener}$ is then at right angles to the local x-axis and z-axis so that the three local axes form a right-handed set.

In the rare cases when the default definition of $\mathbf{T}_{fastener}$ does not suit your application, you can always specify the orientation directly.

Common modeling practices

In most applications the default choice for $\mathbf{T}_{fastener}$ combined with a choice of global system for $\mathbf{T}_{connector}$ at both connector nodes would result in a $\mathbf{T}_{connectorfinal}$ that is most suitable. The

connection type that you choose depends on several modeling considerations, but very often the BUSHING connection type offers the best choice. To simplify the discussion, consider that only two surfaces are being fastened, a very common situation as illustrated in the spot weld example in "Connector functions for coupled behavior," Section 30.2.4. For this common choice, $\mathbf{T}_{connectorfinal}$ has the local z-axis normal to the closest surface and pointing from the first fastening point (first connector node) toward the second fastening point (second connector node). This choice ensures that for a fastener subjected to a tension load (fastened plates pulled apart) a positive force always develops in the connector along the local z-axis (CTF3) regardless of the choice of coordinates for the positioning point and/or use of unsorted surfaces. Conversely, if a compression load is applied (fastened plates pressed against each other), a negative force develops in the connector.

In most cases, the behavior in the tangential plane defined by the local x- and local y-axes is isotropic; therefore, the precise orientation of these two axes is of less interest to you. The spot weld example in "Connector functions for coupled behavior," Section 30.2.4, illustrates such a typical case where the (isotropic) magnitude of two in-plane forces (f_1, f_2) and of the two moments (m_1, m_2) are used in the kinetic behavior of the connector element.

If you need to specify anisotropic behavior in the tangential plane, you need to understand precisely how the directions in $\mathbf{T}_{fastener}$ are defined. As explained above, the choice of coordinates for the positioning point relative to the stack of surfaces to be fastened and/or use of unsorted surfaces determines the precise direction of the default local axes. In most cases you have two common modeling choices. In the first case you can specify the coordinates of the positioning points to be exactly on or very close to the surface onto which the first fastening points (connector nodes) are to be placed and use the default sorted surfaces. In this case you do not need to specify the surfaces to be fastened individually. However, in many practical situations imprecise geometry for the surfaces to be fastened and/or inexact coordinates of the fastener reference nodes make the consistent placement of the reference nodes in the vicinity of one particular surface very hard to accomplish. The second modeling technique consists of using sorted surfaces. The exact location of the reference node with respect to the surface stack to be fastened is not that important because the first fastening point is always on the first specified surface. In this case you do have to specify two or more individual surfaces to be fastened. In the rare cases when neither of these modeling techniques suits your application, you can specify the fastener orientation directly to match your needs exactly.

Defining the surface coupling method

There are two methods available to couple the motion of each fastening point to the motion of the associated coupling nodes on the fastened surfaces: the continuum coupling method and the structural coupling method. The continuum coupling method is used by default.

In many cases when the pair of fastened surfaces are close to each other, unrealistic contact interactions may occur between the two surfaces if the continuum coupling method is used. This is particularly the case in shell bending applications. Moreover, in many situations the continuum coupling method can yield an overly stiff response if the two surfaces are pried apart, especially when the fastener radius is small. The structural coupling method can be used to alleviate these issues.

Continuum coupling method

The default continuum coupling method couples the translation and rotation of each fastening point to the average translation of the group of coupling nodes on each of the fastened surfaces. The constraint distributes the forces and moments at the fastening point as a coupling node-force distribution only. The force distribution is equivalent to the classic bolt pattern force distribution when the weight factors are interpreted as bolt cross-section areas. For each pair of fastening point and group of coupling nodes, the constraint enforces a rigid beam connection between the fastening point and a point located at the weighted center of position of the coupling nodes. The formulation is discussed in detail in "Distributing coupling elements," Section 3.9.8 of the Abaqus Theory Manual.

Input File Usage: *FASTENER, COUPLING=CONTINUUM

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: **Formulation**: **Coupling type**: **Continuum distributing**

Structural coupling method

The structural coupling method couples the translation and rotation of each fastening point to the translation and the rotation motion of the group of coupling nodes on each of the fastened surfaces. The constraint distributes forces and moments at the fastening point as coupling nodes forces and moments. For this coupling method to be active, all rotation degrees of freedom at all coupling nodes must be active (as would be the case when shells are fastened together) and all degrees of freedom must be constrained (which is the default; see "Defining fastener properties" below).

With respect to translations, for each pair of fastening point and group of coupling nodes, the constraint enforces a rigid beam connection between the fastening point and a moving point that remains at all times in the vicinity of the fastened surface. The location of this moving point is determined by the current curvature of the surface, the current location of the weighted center of position of the coupling nodes, and the fastener projection direction. This choice avoids unrealistic contact interactions between the fastened surfaces when the surfaces are close to each other (typically the case).

With respect to rotations, for each pair of fastening point and group of coupling nodes, the constraint is different along different local directions. Along the projection direction (the twist direction), the constraint is identical to the one enforced via the continuum coupling method (see "Distributing coupling elements," Section 3.9.8 of the Abaqus Theory Manual). By contrast, the rotational constraint in the plane perpendicular to the projection direction relates the in-plane fastening point rotations to the in-plane rotations of the coupling nodes in the immediate vicinity of the fastening point. This choice provides a more realistic response when the fastened surfaces are pried apart.

Input File Usage: *FASTENER, COUPLING=STRUCTURAL

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: **Formulation**: **Coupling type**: **Structural distributing**

Defining fastener properties

Each fastener interaction definition must refer to a property, which defines the geometric section properties of the fastener.

Input File Usage: Use both of the following options:

*FASTENER, PROPERTY=*fastener property name*
*FASTENER PROPERTY, NAME=*fastener property name*

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: **Property**

Geometric section quantities

Fasteners are assumed to have a circular projection onto the connected surfaces. You are required to specify the radius of the fastener.

Input File Usage: *FASTENER PROPERTY
r

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: **Property**: **Physical radius**: *r*

Mass

In many cases fasteners may add mass to the assembly. To model the added mass, specify an additional mass that is assigned to each fastener and lumped to the fastening points.

Input File Usage: *FASTENER PROPERTY, MASS=*mass value*

Abaqus/CAE Usage: Interaction module: **Special→Fasteners→Create**: **Point-based**: **Property**: **Additional mass**: *mass value*

Releasing degrees of freedom on fasteners using connector elements

For fasteners modeled with connector elements, translational as well as rotational degrees of freedom can be released by prescribing connector section types that have unconstrained (available) degrees of freedom. For example, a HINGE connector can be used to release the rotational degree of freedom in the connector's local 1-direction.

Releasing degrees of freedom on fasteners using BEAM MPCs

For fasteners modeled with BEAM MPCs, the moment constraint between the rotation degrees of freedom at the fastening points and the average rotation of the coupling nodes can be released in one, two, or three directions. You can specify the moment constraint directions in the default local coordinate system or a user-defined local coordinate system. The three translational degrees of freedom at the fastening points are always coupled to the average translation of the coupling nodes. You specify the degrees of freedom of the fastening point to be coupled to the average motion of the coupling nodes as part of the fastener property definition.

If no degrees of freedom are specified as part of the fastener property definition, all six degrees of freedom are coupled. If you specify one or more degrees of freedom but not all available translation degrees of freedom, Abaqus issues a warning message and adds all the available translation degrees of freedom to the constraint. If a user-specified local orientation is specified for the fastener interaction, the local degrees of freedom are with respect to the user-defined coordinate system.

Input File Usage: ∗FASTENER PROPERTY
section properties
first dof, *last dof*

For example, if the default local coordinate system is used, the following property definition would release the relative rotation constraint of the connected parts about the surface normal:

```
*FASTENER PROPERTY
section properties
1, 5
```

The above property definition might be used to approximate a riveted connection.

Abaqus/CAE Usage: Abaqus/CAE always constrains all translational degrees of freedom in a fastener. Use the following input to remove constraints on the rotational degrees of freedom:

Interaction module: **Special→Fasteners→Create**: **Point-based**: **Formulation**: toggle off **UR1**, **UR2**, or **UR3**

Overconstraints in fasteners modeled with BEAM MPCs

There are several instances in which a model with fasteners modeled with BEAM MPCs might be overconstrained. Described below are two potential overconstraints that Abaqus automatically attempts to detect and resolve during solver input file processing.

Fasteners and rigid bodies

Fasteners can be used to connect both deformable and rigid element-based surfaces. However, if the fasteners are modeled with BEAM MPCs, potential overconstraints may arise if more than one rigid surface is involved in a given fastener definition. Abaqus automatically attempts to remove these types of overconstraints by allowing at most one rigid surface in any individual fastener definition. A warning message is generated if an overconstraint of this type is detected.

For example, suppose surfaces A and C in Figure 33.3.4–1 are part of the same rigid body, and surface B is deformable. Abaqus automatically removes either surface A or surface C from the fastener definition and only forms the fastener between the deformable surface and the remaining rigid surface. If surface A and surface C belong to two separate rigid bodies, their respective rigid body reference nodes will be joined by an internally generated BEAM MPC.

In another example, suppose all three surfaces in Figure 33.3.4–1 are rigid. In this case no fastener will be formed, and the unique rigid body reference nodes for surfaces A, B, and C will be joined by beam MPCs. Unresolvable overconstraints may arise if inconsistent kinematic constraints (such as

displacement boundary conditions) are placed on rigid body reference nodes that have been joined by BEAM MPCs. In this case you must modify the model to resolve the overconstraints. Possible courses of action include removing some of the rigid surfaces from the fastener definitions or removing inconsistent kinematic conditions on the rigid body reference nodes.

The above-described procedure to resolve overconstraints with fasteners and rigid bodies will preserve the kinematics of the original model. In Abaqus/Standard you can bypass the overconstraint checks and prevent automatic model modifications in the model preprocessor (see "Overconstraint checks," Section 33.6.1).

Overlapping fasteners

Potential overconstraints exist with rigid fasteners if all the coupling nodes of any associated distributing coupling element are wholly contained within one or more other fastener definitions. This can happen if the spacing between positioning points is small compared to the typical element size in a mesh (which is often the case in automotive models). To avoid overconstraints in this situation, Abaqus uses a penalty formulation for all fastener distributing coupling elements that satisfy the above criteria. The penalty distributing coupling formulation relaxes, to a small degree, the constraint between the motion of the distributing coupling element reference node and its coupling nodes.

Output

If fasteners are modeled using connector elements, connector element output variables can be used to request output for fasteners (see "Connector elements," Section 30.1.2). No fastener output is available if the fasteners are modeled using BEAM MPCs.

33.4 Embedded elements

- “Embedded elements,” Section 33.4.1

33.4.1 EMBEDDED ELEMENTS

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Kinematic constraints: overview," Section 33.1.1
- *EMBEDDED ELEMENT
- "Defining embedded region constraints," Section 15.15.8 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

The embedded element technique:

- is used to specify an element or a group of elements that lie embedded in a group of host elements whose response will be used to constrain the translational degrees of freedom of the embedded nodes (i.e., nodes of embedded elements);
- can be used in geometrically linear or nonlinear analysis;
- is not available for host elements with rotational degrees of freedom;
- can be used to model a set of rebar-reinforced membrane, shell, or surface elements that lie embedded in a set of three-dimensional solid (continuum) elements; a set of truss or beam elements that lie embedded in a set of solid elements; or a set of solid elements that lie embedded in another set of solid elements;
- will not constrain rotational degrees of freedom of the embedded nodes when shell or beam elements are embedded in solid elements; and
- can be imported from Abaqus/Standard into Abaqus/Explicit and vice versa.

Introduction

The embedded element technique is used to specify that an element or group of elements is embedded in "host" elements. The embedded element technique can be used to model rebar reinforcement. Abaqus searches for the geometric relationships between nodes of the embedded elements and the host elements. If a node of an embedded element lies within a host element, the translational degrees of freedom at the node are eliminated and the node becomes an "embedded node." The translational degrees of freedom of the embedded node are constrained to the interpolated values of the corresponding degrees of freedom of the host element. Embedded elements are allowed to have rotational degrees of freedom, but these rotations are not constrained by the embedding. Multiple embedded element definitions are allowed.

Available embedded element types

Different element types can be used in the element set containing embedded elements and the element set containing the host elements. However, all the host elements can have only translational degrees of freedom, and the number of translational degrees of freedom at a node on the embedded element must be identical to the number of translational degrees of freedom at a node on the host element. The following general types of "embedded elements-in-host elements" are provided:

- Two-dimensional models:
  - Beam-in-solid
  - Solid-in-solid
  - Truss-in-solid
- Axisymmetric models:
  - Membrane-in-solid (Abaqus/Standard only)
  - Shell-in-solid
  - Solid-in-solid
  - Surface-in-solid (Abaqus/Standard only)
- Three-dimensional models:
  - Beam-in-solid
  - Membrane-in-solid
  - Shell-in-solid
  - Solid-in-solid
  - Surface-in-solid
  - Truss-in-solid

Specifying the host elements

By default, the elements in the vicinity of the embedded elements are searched for elements that contain embedded nodes; the embedded nodes are then constrained by the response of these host elements. To preclude certain elements from constraining the embedded nodes, you can define a host element set; the search will be limited to this subset of the host elements in the model. This feature is strongly recommended if the embedded nodes are close to discontinuities in the model (cracks, contact pairs, etc.).

Input File Usage: *EMBEDDED ELEMENT, HOST ELSET=*name*

The *EMBEDDED ELEMENT option must be included in the model definition portion of the input file. Multiple *EMBEDDED ELEMENT options are allowed.

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Embedded region**: choose **Select Region** from the prompt area when selecting the host region

Specifying the embedded elements

You must specify the embedded elements. Individual elements or element sets can be specified.

An embedded element may share some nodes with host elements. These nodes, however, will not be considered to be embedded nodes.

Input File Usage: *EMBEDDED ELEMENT
embedded elements

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Embedded region**: select the embedded region

Defining geometric tolerances

A geometric tolerance is used to define how far an embedded node can lie outside the regions of the host elements in the model. By default, embedded nodes must lie within a distance calculated by multiplying the average size of all non-embedded elements in the model by 0.05; however, you can change this tolerance.

You can define the geometric tolerance as a fraction of the average size of all non-embedded elements in the model. Alternatively, you can define the geometric tolerance as an absolute distance in the length units chosen for the model. If you specify both exterior tolerances, Abaqus uses the tighter tolerance of the two. The average size of all the non-embedded elements is calculated and multiplied by the fractional exterior, which is then compared to the absolute exterior tolerance to determine the tighter tolerance of the two. The exterior tolerance for embedded elements in host elements is indicated by the shaded region in Figure 33.4.1–1.

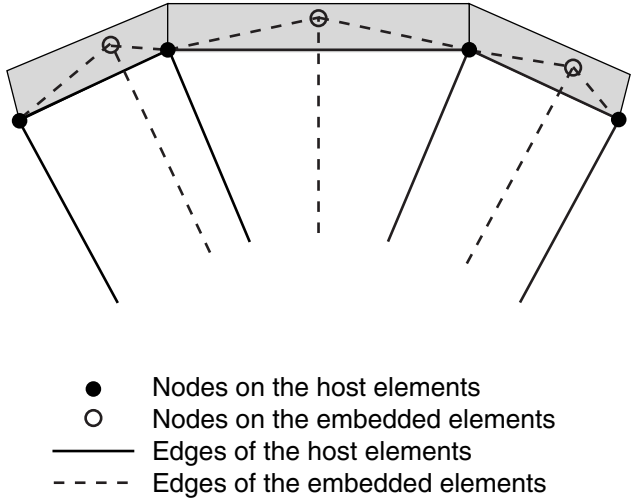


Figure 33.4.1–1 The exterior tolerance for embedded elements.

If an embedded node is located inside the specified tolerance zone, the node is constrained to the host elements. The position of this node will be adjusted to move the node precisely onto the host elements. If an embedded node is located outside the specified tolerance zone, an error message will be issued.

Input File Usage: Use the following option to define the tolerance as a fraction:

*EMBEDDED ELEMENT, EXTERIOR TOLERANCE=*tolerance*

Use the following option to define the tolerance as an absolute distance:

*EMBEDDED ELEMENT,
ABSOLUTE EXTERIOR TOLERANCE=*tolerance*

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Embedded region**: **Fractional exterior tolerance** or **Absolute exterior tolerance**

Adjusting the positions of embedded nodes

If an embedded node lies close to an element edge or an element face within a host element, it is computationally efficient to make a small adjustment to the position of the embedded node so that the node will lie precisely on the edge or face of the host element. A small tolerance, below which the weight factors of the nodes on a host element associated with an embedded node will be zeroed out, is defined. The small weight factors will be redistributed to the other nodes on the host element in proportion to their initial weights, and the position of the embedded node will be adjusted based on the new weight factors. This adjustment is performed only at the start of the analysis and does not create any strain in the model. It is most useful for making small adjustments to make the embedded nodes lie on the edge or face of a host element. If a large nondefault value of the roundoff tolerance is used to make significant adjustments to the positions of the embedded nodes, you should carefully review the mesh obtained after adjusting.

Input File Usage: *EMBEDDED ELEMENT, ROUNDOFF TOLERANCE=*tolerance*

Abaqus/CAE Usage: Interaction module: **Create Constraint**: **Embedded region**: **Weight factor roundoff tolerance**

Use with other multiple kinematic constraints

If an embedded node is also tied by multi-point, equation, kinematic coupling, surface-based tie, or rigid body constraints, an overconstraint is introduced and an error message will be issued. If a boundary condition is applied to an embedded node, the embedded element definition always takes precedence. The boundary condition will be neglected, and a warning message will be issued.

Defining surfaces on embedded elements

Embedded elements have no exterior (free) surface due to the embedding. Consequently, their faces are not part of the all-inclusive surface defined automatically for interactions modeled with general contact. In addition, any surface definitions based on these elements must have the face identifier specified explicitly (see “Element-based surface definition,” Section 2.3.2).

Limitations

The following limitations exist for the embedded element technique:

- Elements with rotational degrees of freedom (except axisymmetric elements with twist) cannot be used as host elements.
- Rotational, temperature, pore pressure, acoustic pressure, and electrical potential degrees of freedom at an embedded node are not constrained.
- Host elements cannot be embedded themselves.
- The material defined for the host element is not replaced by the material defined for the embedded element at the same location of the integration point.
- Additional mass and stiffness due to the embedded elements are added to the model.

Example

Consider the example in Figure 33.4.1–2.

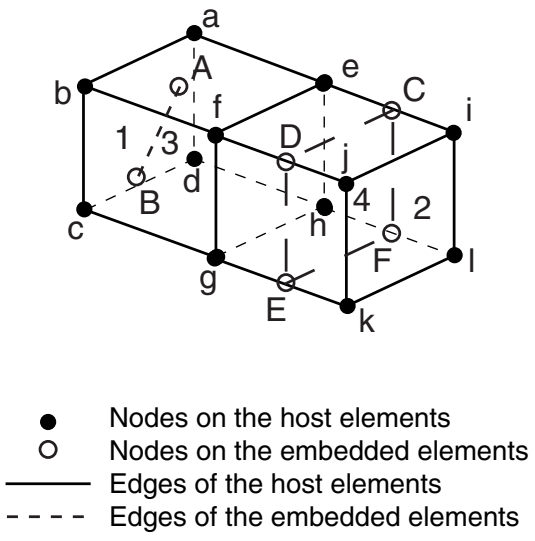


Figure 33.4.1–2 Elements lie embedded in host elements.

Elements 3 (truss) and 4 (membrane) lie embedded in elements 1 and 2. Element 1 is formed by nodes a, b, c, d, e, f, g, and h; element 2 is formed by nodes e, f, g, h, i, j, k, and l; element 3 is formed by nodes A and B; and element 4 is formed by nodes C, D, E, and F. If the host element set includes elements 1 and 2 and the embedded element sets contain elements 3 and 4, respectively, Abaqus will attempt to find if there are any embedded nodes (A, B, C, D, E, and F) lying within host elements 1 or 2. If node A is found to be lying close to the a-b-f-e face of element 1, all the degrees of freedom at node A are constrained to

nodes a, b, f, and e, with appropriate weight factors being determined based on the geometric location of node A in element 1. Similarly, if node B is found to be lying inside element 1 and node E is found to be lying close to the g–k edge of element 2, respectively, all the degrees of freedom at node B are constrained to nodes a, b, c, d, e, f, g, and h, and all the degrees of freedom at node E are constrained to nodes g and k, with appropriate weight factors being determined based on the geometric location of node B in element 1 and the geometric location of node E on the g–k edge of element 2, respectively.

You should make sure that all the nodes on the embedded elements are properly constrained to nodes on the host elements. This can be verified by performing a data check analysis (see "Abaqus/Standard, Abaqus/Explicit, and Abaqus/CFD execution," Section 3.2.2). For each embedded node a list of nodes that are used to constrain this node and the associated weight factors are output to the data file during the data check analysis. An error message is issued if an embedded node is not constrained.

Template

```
*HEADING
...
*NODE
Data line to define the nodal coordinates
*ELEMENT, TYPE=C3D8, ELSET=SOLID3D
Data line to define the solid elements
*ELEMENT, TYPE=T3D2, ELSET=TRUSS
Data line to define the truss elements
*ELEMENT, TYPE=M3D4, ELSET=MEMB
Data line to define the membrane elements
*EMBEDDED ELEMENT, EXTERIOR TOLERANCE=tolerance, HOST ELSET=SOLID3D
TRUSS, MEMB
*STEP
*STATIC (or any other allowable procedure)
Data line to define step time and control incrementation
...
*END STEP
```

33.5 Element end release

- “Element end release,” Section 33.5.1

33.5.1 ELEMENT END RELEASE

Product: Abaqus/Standard

References

- "Kinematic constraints: overview," Section 33.1.1
- *RELEASE

Overview

Element end release:

- allows a rotational degree of freedom or a combination of rotational degrees of freedom to be released at one or both ends of an element or element set;
- can be used in geometrically linear or nonlinear analysis; and
- is available only for beam and pipe elements in Abaqus/Standard.

Introduction

Element end release is used to model hinged connections (hinged in one, two, or three orthogonal directions) at one or both ends of the element. By releasing rotational degrees of freedom, an element end is allowed to rotate freely relative to the node about the chosen degrees of freedom. Any rotational degrees of freedom that are not released are shared with the node. You must be careful not to release a given degree of freedom at a node for all elements that share that node; otherwise, the node has no stiffness for that degree of freedom and Abaqus/Standard issues zero pivot warning messages.

Element end release operates on the element local degrees of freedom. See "Beam element cross-section orientation," Section 28.3.4, for a definition of the local axes (n_1, n_2, t) for beam-type elements. The rotational degrees of freedom affected by the release are the rotation about the local n_1-axis, the rotation about the local n_2-axis, and the rotation about the local t-axis for beams in space. For beams in a plane, only the rotation about the local n_1-axis is active (which coincides with rotations about the negative global z-axis).

Equivalent MPCs

If only one rotational degree of freedom is released, the kinematic constraint is equivalent to MPC type REVOLUTE plus MPC type PIN between two nodes. If two rotational degrees of freedom are released, the kinematic constraint is equivalent to MPC type UNIVERSAL plus MPC type PIN. If all rotational degrees of freedom are released, the kinematic constraint is equivalent to MPC type PIN. See "General multi-point constraints," Section 33.2.2, for details.

Identifying the element end involved in the release

Either element sets or individual elements can be specified for a release definition. Degrees of freedom can be released at the first, second, or first and second ends of an element. The first end of the element, S1, is node 1 on the element as defined by the element connectivity; the second end, S2, is the last node (node 2 or 3, as appropriate) on the element. See "Beam element library," Section 28.3.8, for a definition of the node ordering for beam elements.

Identifying the local rotational degrees of freedom involved in the release

Rotation combination codes rather than degrees of freedom are specified to identify the rotational degrees of freedom involved in the release.

M1 refers to the rotation about the n_1-axis,

M2 refers to the rotation about the n_2-axis,

M1-M2 refers to a combination of rotational degrees of freedom about the n_1-axis and the n_2-axis,

T refers to the rotation about the t-axis,

M1-T refers to a combination of rotational degrees of freedom about the n_1-axis and the t-axis,

M2-T refers to a combination of rotational degrees of freedom about the n_2-axis and the t-axis, and

ALLM represents a combination of all the rotational degrees of freedom (i.e., M1, M2, and T).

Input File Usage: ∗RELEASE
element number or element set, element end ID, release combination code

For example, to release the rotational degree of freedom about the n_1-axis at the first end of element 10 and all the rotational degrees of freedom at the second end of the element, use the following input:

```
*RELEASE
10, S1, M1
10, S2, ALLM
```

Use with transformed coordinate systems

Transformations applied to released nodes ("Transformed coordinate systems," Section 2.1.5) have no influence on the release. The release operates on the local degrees of freedom for the element.

Reading the data from an alternate input file

The data for a release definition can be contained in a separate input file.

Input File Usage: ∗RELEASE, INPUT=*file_name*

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

33.6 Overconstraint checks

- “Overconstraint checks,” Section 33.6.1

33.6.1 OVERCONSTRAINT CHECKS

Product: Abaqus/Standard

References

- "Rigid body definition," Section 2.4.1
- "Connectors: overview," Section 30.1.1
- "Boundary conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.3.1
- "General multi-point constraints," Section 33.2.2
- "Mesh tie constraints," Section 33.3.1
- "Coupling constraints," Section 33.3.2
- "Mesh-independent fasteners," Section 33.3.4
- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- *BASE MOTION
- *CONSTRAINT CONTROLS

Overview

An overconstraint means applying multiple consistent or inconsistent kinematic constraints. Many models have nodal degrees of freedom that are overconstrained. Such overconstraints may lead to inaccurate solutions or nonconvergence. Common examples of situations that may lead to overconstraints include (but are not limited to):

- contact slave nodes that are involved in boundary conditions or multi-point constraints;
- edges of surfaces involved in a surface-based tie constraint that are included in contact slave surfaces or have symmetry boundary conditions; and
- boundary conditions applied to nodes already involved in coupling or rigid body constraints.

The overconstraint checks performed in Abaqus/Standard:

- check for overconstraints caused by combinations of the following: base motions, boundary conditions, contact pairs, coupling constraints, linear constraint equations, mesh-independent spot welds, multi-point constraints, rigid body constraints, and surface-based tie constraints;
- check for overconstraints resulting from kinematic constraints introduced through connector elements, coupling elements, special-purpose contact elements, and elements with incompressible material behavior;
- identify through detailed messages the constraints that cause overconstraints;
- automatically resolve a limited set of consistent overconstraints detected during model preprocessing and during an Abaqus/Standard analysis;

- use the equation solver to detect overconstraints that cannot be resolved automatically; and
- can have the default behavior modified.

Overconstraints: general remarks

In general, the term overconstraint refers to multiple constraints acting on the same degree of freedom. Overconstraints are then categorized as *consistent* (if all the constraints are compatible with each other) or *inconsistent* (if the constraints are incompatible with each other). Consistent overconstraints are also called *redundant* constraints, and inconsistent overconstraints are also called *conflicting* constraints.

In Abaqus/Standard the following types of constraints, in combination, may lead to overconstraints:

- boundary conditions or base motions,
- contact pairs,
- coupling constraints,
- mesh-independent spot welds,
- multi-point constraints or linear constraint equations,
- surface-based tie constraints, and
- rigid body constraints.

In addition to these constraints the following elements impose kinematic constraints and, when used in combination with each other or with the above constraints, may lead to overconstraints:

- connector elements,
- special-purpose contact elements, and
- hybrid elements for incompressible material response.

An illustration of several consistent overconstraints is given in Figure 33.6.1–1. The upper block is built from three separately meshed regions, which are connected together using a surface-based tie constraint. This block is in contact with the lower rigid block, which is made rigid by specifying a rigid body constraint. The rigid block's reference node is fixed. Symmetry boundary conditions are used at the left edge of the upper block, and rough friction is defined for the surface interaction between the upper and lower blocks. The following redundant constraints can be identified:

- Intersecting tie constraints: At (A) three nodes share the same location, and their relative motions are constrained by two surface-based tie constraints (one vertical and one horizontal). Only two constraints (two dependent nodes and one independent node) are needed to fully constrain the motion of the three nodes, but three constraints are generated internally (one for the horizontal tie constraint and two for the vertical one). Therefore, one redundant constraint exists.
- Tie constraint and symmetry boundary condition: At (B) nodes 141 and 151 have their motion constrained horizontally by the symmetry boundary condition, but their relative motion is also constrained by the surface-based tie constraint. Therefore, one redundant constraint exists.
- Rough friction and symmetry boundary condition: At (C) node 101 is constrained horizontally by the symmetry boundary condition. The rough friction contact acts in the same direction as the boundary condition. Therefore, one redundant constraint exists.

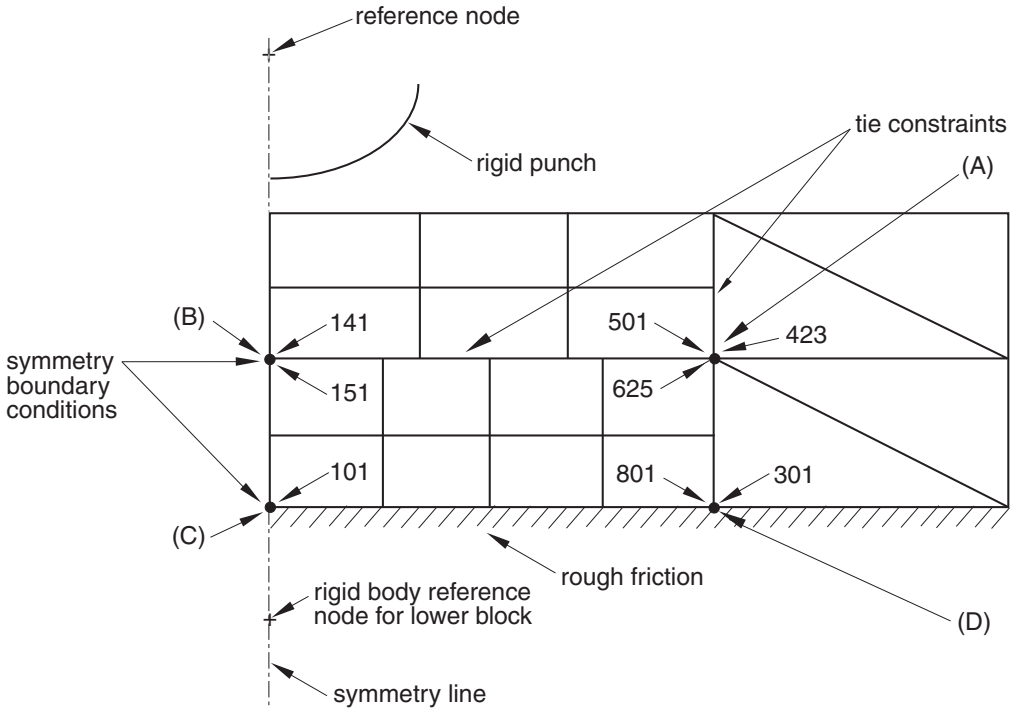


Figure 33.6.1–1 Model with redundant constraints.

- Tie constraint and contact interactions: At (D) nodes 801 and 301 are involved in the surface-based tie constraint, but two contact constraints (one at each node) act in the vertical direction. Therefore, one redundant constraint exists.

Even in this simple model the number of redundant constraints is surprisingly large. If not appropriately accounted for, the redundant constraints can lead to convergence difficulties, even nonconvergence. Moreover, in the cases when a solution is obtained (despite the convergence difficulties), the reported reaction forces and contact pressures may be inaccurate.

Abaqus/Standard checks for the inappropriate use of combinations of constraints for the majority of constraint and element types listed in this section. Depending on the complexity of the constraints involved, Abaqus/Standard identifies three classes of consistent and inconsistent overconstraints.

Overconstraints detected in the model preprocessor

Many relatively simple overconstraints can be identified by inspecting the constraints defined at a node. If a consistent overconstraint is detected, the unnecessary constraints are eliminated automatically and a warning message is generated. If the overconstraints are inconsistent, the analysis is stopped and an error message is generated.

Overconstraints detected and resolved in an Abaqus/Standard analysis

Some overconstraints involving contact interactions may become overconstrained only during an analysis due to changes in contact status. Certain of these cases are detectable and eliminated automatically by Abaqus/Standard. Appropriate messages are issued.

Overconstraints detected by the equation solver

Many overconstraints involve complex interactions between various constraint definitions and element types. Automatic resolution of these situations may not be possible. In such cases the equation solver will detect the overconstraint, and a detailed message listing potential causes of the problem will be issued.

Overconstraints detected in the model preprocessor

In this section we consider overconstraints that involve two or more of the following:

- surface-based tie constraints,
- rigid body constraints,
- boundary conditions, and
- connector elements.

While the number of cases handled automatically in the model preprocessor is limited, many often-encountered situations are corrected. The list of overconstraints to be resolved automatically in the preprocessor is organized based on the constraint types involved. Each case is illustrated by examples.

Intersecting tie constraints

Examples of intersecting tie constraint definitions are shown in Figure 33.6.1–2. In both cases there is at least one node that, if not properly treated, will be redundantly constrained. In the case on the left, the three edges belonging to the three surfaces overlap (shown here in an exploded view for clarity). Each of the three end nodes on either end occupy the same location. Therefore, one redundant tie constraint exists. In the case shown on the right, four adjacent meshes are "glued" together using four tie constraints. Only three constraints are needed to "glue" the center nodes together, but four are generated (one from each tie constraint). Therefore, one constraint is not needed and in both cases one constraint is removed.

Tie constraint inside a rigid body constraint

An example of a tie constraint inside a rigid body constraint is shown in Figure 33.6.1–3(a). Two surfaces are connected by a tie constraint, and the two element sets are included in the same rigid body. Since the motion of all the nodes is constrained to the motion of the rigid body's reference node, the tie constraint is redundant. The tie constraint definition is removed from the model.

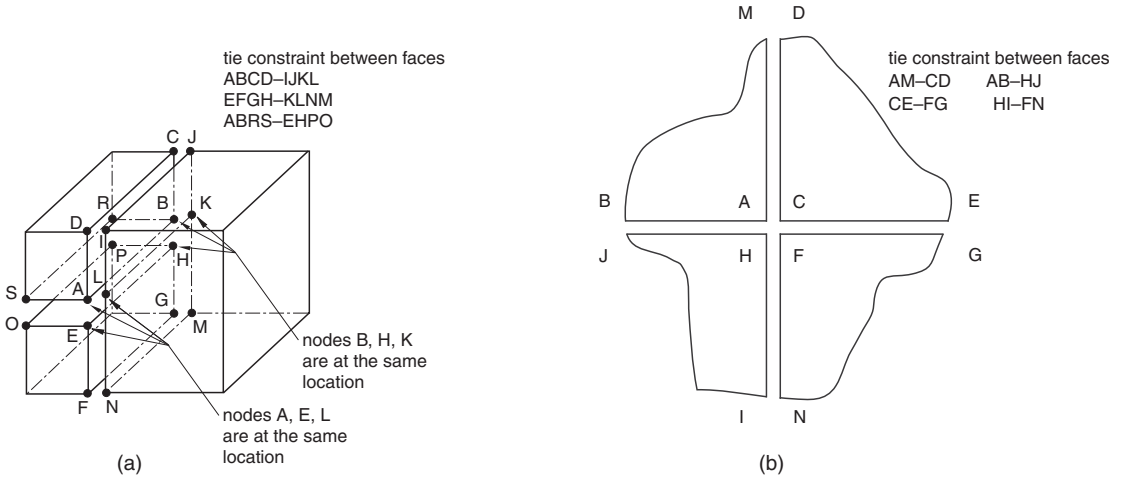


Figure 33.6.1–2 Consistent overconstraints due to intersecting tie constraints.

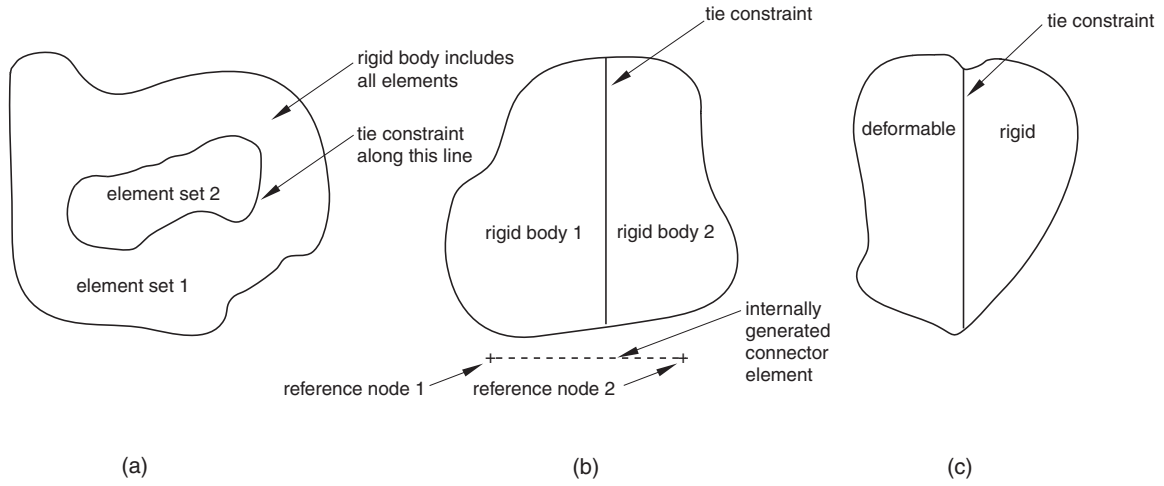


Figure 33.6.1–3 Consistent overconstraints due to combinations of tie and rigid body constraints.

Tie constraint between two rigid bodies

An example of a tie constraint between two rigid bodies is shown in Figure 33.6.1–3(b). If the two surfaces are connected by a tie constraint at more than two or three points (in two- or three-dimensional analyses, respectively), the tie constraint definition is redundant. A connector type BEAM is placed between the two reference nodes, and the tie constraint is removed.

Tie constraint between a deformable and a rigid body

An example of connecting a deformable body to a rigid body with a surface-based tie constraint is shown in Figure 33.6.1–3(c). If the slave surface in the tie constraint definition belongs to the rigid body, the tie and the rigid body constraints are redundant for the slave nodes. If possible, Abaqus/Standard will switch the master and the slave surface in the tie constraint definition. If switching the master and the slave surfaces is not possible due to other modeling restrictions, an error message is issued and the analysis is stopped.

Intersecting rigid bodies

Figure 33.6.1–4(a) illustrates the case when two rigid bodies partially overlap and, thus, the union of the two bodies behaves as one rigid body. However, the motion of the nodes in this region is governed by the motion of the two rigid body reference nodes; hence, the model is overconstrained. In Figure 33.6.1–4(b) several rigid bodies are included in a larger rigid body definition. The nodes belonging to the included bodies will be overconstrained.

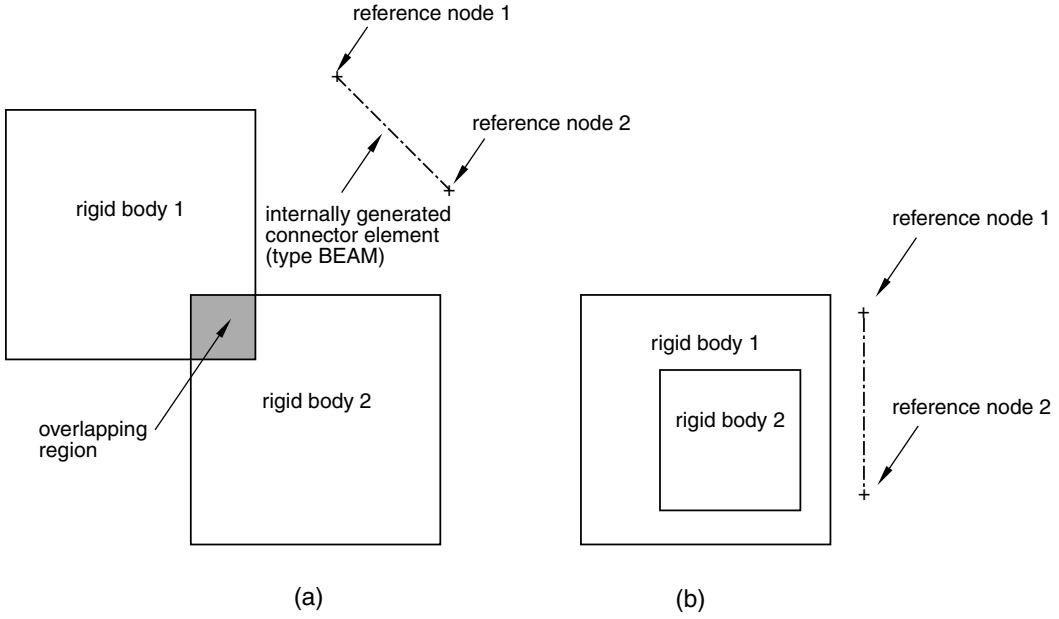


Figure 33.6.1–4 Rigid body including other rigid bodies.

In both cases the rigid body constraint will be enforced only once for the nodes that belong to several rigid bodies. To enforce the rigid behavior of the ensemble, connector elements of type BEAM are generated between the rigid body reference nodes to ensure a rigid connection between the intersecting rigid body definitions.

Tie constraints and boundary conditions

There are numerous cases of overconstraints when a surface-based tie constraint and a boundary condition are used together, as illustrated in Figure 33.6.1–5.

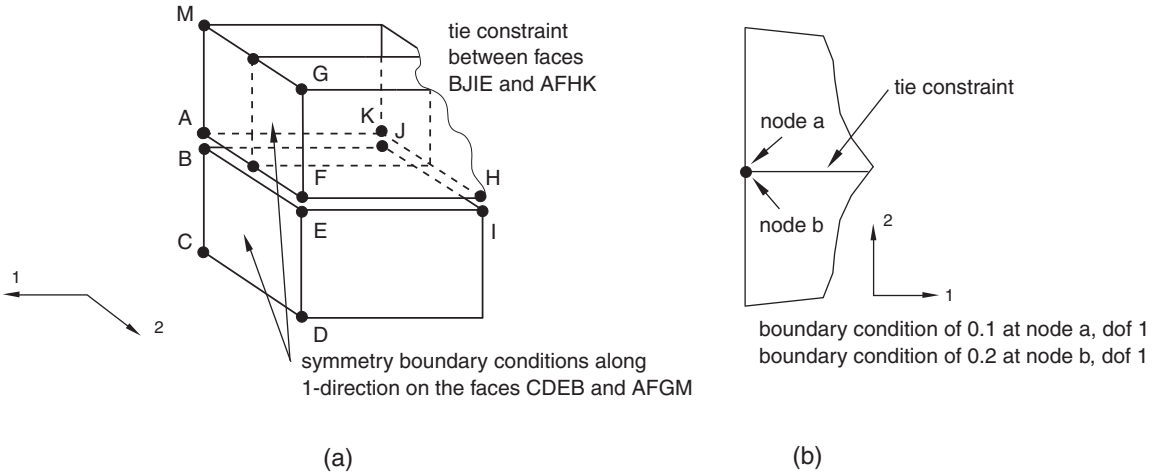


Figure 33.6.1–5 Overconstraints involving tie constraints and boundary conditions.

In the first case nodes A and B are constrained to move together by the tie constraint. The vertical symmetry boundary conditions will constrain the motion of both nodes in the horizontal direction, generating one redundant constraint. In the second case the two specified boundary conditions conflict, thus generating a conflicting constraint.

For every tie-dependent node with a boundary condition, Abaqus/Standard first determines which independent nodes are involved in the tie constraint (see "Mesh tie constraints," Section 33.3.1). If only one independent node is involved, Abaqus/Standard will transfer the boundary conditions from the dependent node to the independent node. If conflicting boundary conditions are detected at the independent node during the transferring process, the analysis is stopped and an error message is issued. If several independent nodes are involved, Abaqus/Standard checks if the specified boundary conditions at all the nodes involved in the constraint are identical. If no conflicts are identified, the boundary conditions at the independent node are redundant and, therefore, ignored. Otherwise, an error message is issued, and the analysis is stopped.

Rigid body constraints and boundary conditions

Combinations of rigid body constraints and boundary conditions can lead to overconstrained models when boundary conditions are specified at nodes other than the reference node (Figure 33.6.1–6). In Figure 33.6.1–6(a) boundary conditions are specified at several nodes belonging to the rigid body. In Figure 33.6.1–6(b) symmetry boundary conditions are specified on the flat surface of the rigid body, and the body is spun around an axis perpendicular to the symmetry plane at the reference node.

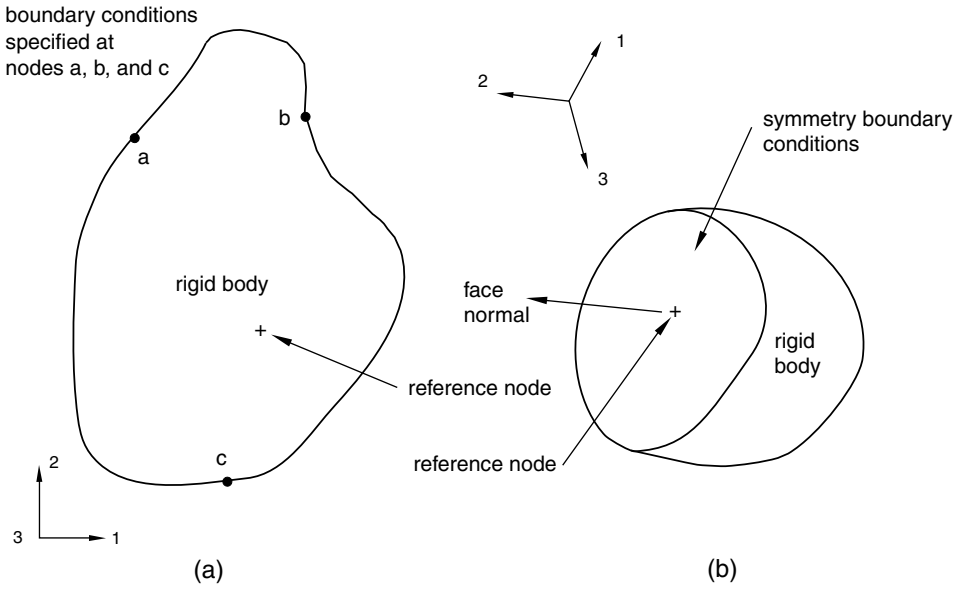


Figure 33.6.1–6 Overconstraints due to boundary conditions applied at rigid body nodes.

In case (a) if the specified boundary conditions are not consistent with the rigid constraint, the model will be inconsistently overconstrained. In case (b) if the reference node has the symmetry boundary conditions, there is no need to have symmetry boundary conditions at the nodes of the flat surface. Abaqus/Standard will attempt to remove all boundary conditions specified at the dependent nodes and redefine them at the reference node. To do so, the consistency of the boundary conditions specified at the dependent nodes is checked. If the boundary conditions are not identical, an error message is issued and the analysis is stopped (since otherwise the solution of a nonlinear system of equations would be required in the general case to assess whether the boundary conditions are consistent or not). Otherwise, Abaqus/Standard will try to merge the boundary conditions at the dependent nodes with those at the reference node by:

- checking the consistency of the overlapping boundary conditions;
- moving to the reference node any boundary conditions specified at the dependent nodes but not specified at the reference node; and
- applying additional zero rotational boundary conditions at the reference node to compensate for the removed displacement constraints from the dependent nodes.

To illustrate, refer to Figure 33.6.1–6(b): as the symmetry boundary conditions specified at the dependent nodes are consistent with each other, they are removed from the dependent nodes and applied to the reference node (boundary condition in the 2-direction). In addition, the symmetry constraints preclude

rotations about the 1- and 3-directions; therefore, zero rotational boundary conditions are applied to the reference node about these axes.

Connector elements and rigid bodies

In most cases detection and automatic resolution of redundant constraints involving connector elements cannot be done by simple inspection of the constraints involved. However, the examples shown in Figure 33.6.1–7 are simple enough to be resolved automatically. It is assumed that the connector elements are connected to nodes on the rigid body whose rotational degrees of freedom are dependent on the rotation of the reference node. In Figure 33.6.1–7(a) the connector elements are assumed to enforce some kinematic constraints. They are redundant since the rigid body definition constrains the motion of all nodes to the motion of the rigid body's reference node. Abaqus/Standard automatically removes the connector elements from the model.

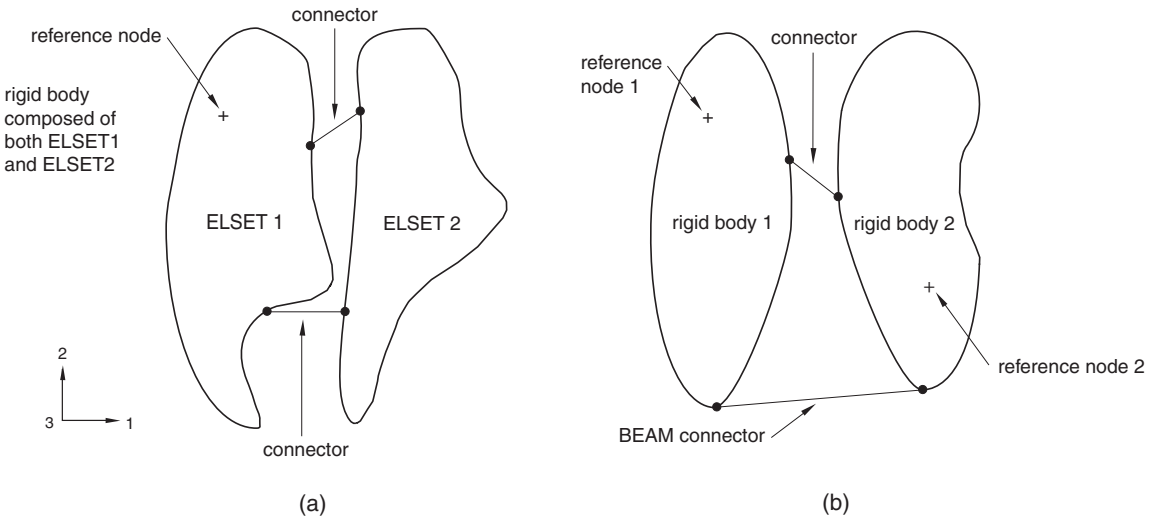


Figure 33.6.1–7 Redundant constraints involving rigid bodies and connector elements.

When connector elements are placed between two rigid bodies (as in Figure 33.6.1–7(b)), the model may be redundantly constrained. As shown in Figure 33.6.1–7(b), if a connector element of type BEAM (or WELD) is placed between two rigid bodies, the connection is rigid and any additional connector elements between the two rigid bodies are redundant. Abaqus/Standard will automatically remove these redundant connector elements.

When the ensemble of connector elements placed between two rigid bodies enforces more than the necessary translational and rotational constraints between the two rigid bodies, but none of the connectors is of type BEAM (or WELD), only warning messages are issued to signal the overconstraint situation. In these cases none of the connector elements can be eliminated automatically since the

connection between the two rigid bodies may become underconstrained. To illustrate this situation, assume that in Figure 33.6.1–7(b) the two connectors were of type SLOT and TRANSLATOR. Thus, four translational constraints (in three dimensions) are enforced between the two rigid bodies, rendering the system overconstrained since only three translational constraints are needed to fully constrain the relative translation between the two bodies. However, if the SLOT were eliminated from the model, the model would become underconstrained and different from the original one. Only a warning message is issued in this case.

Coupling constraints and rigid bodies

When all or some of the nodes involved in a kinematic coupling constraint belong to the same rigid body, the coupling constraint becomes redundant. The situation is illustrated in Figure 33.6.1–8. Node 101 is the reference node for the coupling constraint involving nodes 1001–1005. At the same time nodes 1001–1003 are included in the rigid body definition with reference node 102.

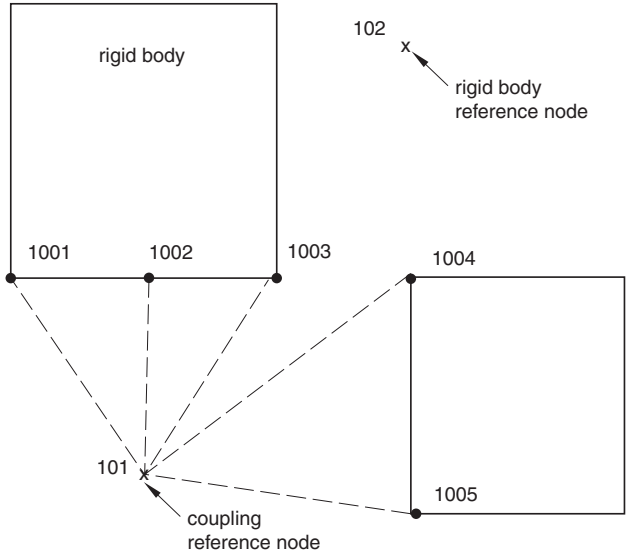


Figure 33.6.1–8 Redundant constraints involving coupling constraints and rigid bodies.

If the coupling constraint was defined as kinematic, it will not be enforced at nodes 1001–1003 to avoid overconstraining the model. The removed overconstraint may be inconsistent such as when incompatible boundary conditions are prescribed at the two reference nodes. However, the constraint will be enforced at nodes 1004 and 1005 since these nodes do not belong to the rigid body.

If a distributing coupling constraint was used instead, the model would not be overconstrained. However, if node 101 was added to the rigid body definition and nodes 1004 and 1005 were not included in the coupling constraint, the model would be overconstrained. Indeed, all nodes involved in the coupling constraint would be already constrained by the rigid body definition, making the coupling

constraint redundant. To avoid the overconstraint, Abaqus/Standard will not enforce the coupling constraint in this case.

Coupling constraints and boundary conditions

When boundary conditions are specified at all nodes involved in a distributing coupling constraint, the model may become overconstrained. Abaqus/Standard will issue a warning message outlining the cause of the potential overconstraint.

Spot welds and rigid bodies

Potential overconstraints that may arise when a rigid body is involved in a mesh-independent spot weld definition are discussed in "Mesh-independent fasteners," Section 33.3.4.

Overconstraints detected and resolved during analysis

There are numerous situations when contact interactions in combination with other constraint types may lead to overconstraints. Since contact status typically changes during the analysis, it is not possible to detect redundant constraints associated with contact in the model preprocessor. Instead, these checks are performed during the analysis. Due to the complexities associated with contact interactions, only a limited number of redundant constraint cases are resolved automatically.

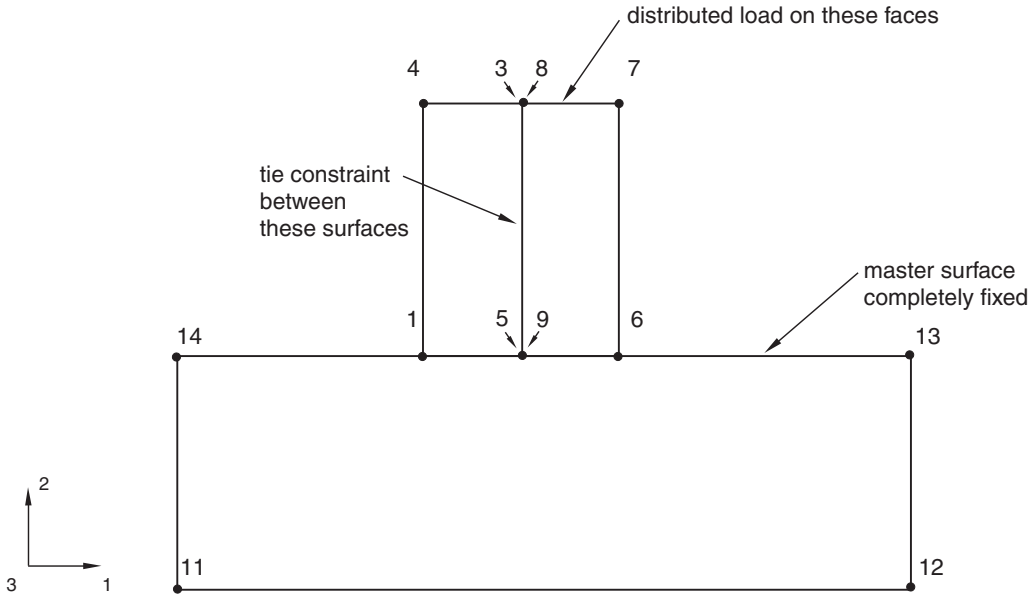
Contact interactions and tie constraints

Redundant constraints are common in cases when slave nodes used in surface-based tie constraints ("Mesh tie constraints," Section 33.3.1) are also slave nodes in contact, as illustrated in Figure 33.6.1–9. In Figure 33.6.1–9(a) nodes 5 and 9 are connected with a tie constraint, and both are in contact with a master surface. Since the two nodes are tied together, one of the contact constraints is redundant. A similar situation is presented in Figure 33.6.1–9(b): two mismatched solid meshes are connected with a tie constraint, and contact is defined with a flat rigid surface. Node S is a dependent node in the tie constraint, so its motion is determined by that of nodes B and C. Therefore, any contact constraint applied at node S is redundant. Moreover, the contact constraints at nodes G and H are redundant, since the motion of these nodes is determined by nodes B and C, respectively. To eliminate these redundancies when all nodes involved in the tie constraint are in contact, Abaqus/Standard will automatically apply a tie-type constraint between the Lagrange multipliers associated with the contact constraint. The redundant contact constraint is eliminated. The contact pressure and the friction forces at the slave node are recovered from the pressures and friction forces at the associated tie-independent nodes.

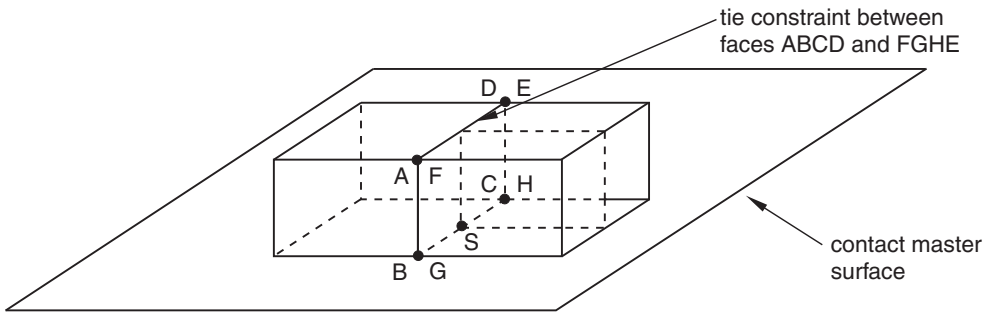
Deleting contact elements to remove overconstraints

Instead of letting Abaqus remove overconstraints by tying Lagrange multipliers, you can apply constraint controls that delete the contact elements associated with tied slave nodes. If you use this technique, contact-related output is not available for the tied slave nodes.

Input File Usage: *CONSTRAINT CONTROLS, DELETE SLAVE



(a)



(b)

Figure 33.6.1–9 Redundant constraints arising from contact interactions and tie constraints.

Contact interactions and prescribed boundary conditions

Contact interactions and prescribed boundary conditions may lead to redundant constraints if either normal contact with the default "hard contact" formulation ("Contact pressure-overclosure

relationships," Section 35.1.2) or frictional contact with the Lagrange multiplier formulation (see "Frictional behavior," Section 35.1.5) is invoked. Abaqus/Standard attempts to resolve these types of redundant constraints for contact pairs involving rigid surfaces.

Checks related to normal contact interactions

In Figure 33.6.1–10 the fixed analytical rigid master surface is in contact with a slave node that has a fixed boundary condition specified in the direction normal to the contact surface. If during a particular increment in the analysis the node is in contact, the contact constraint is redundant and will not be enforced during that increment. If the boundary condition at the slave node is in conflict with the boundary conditions at the rigid surface's reference node, an error message is issued and the analysis is stopped.

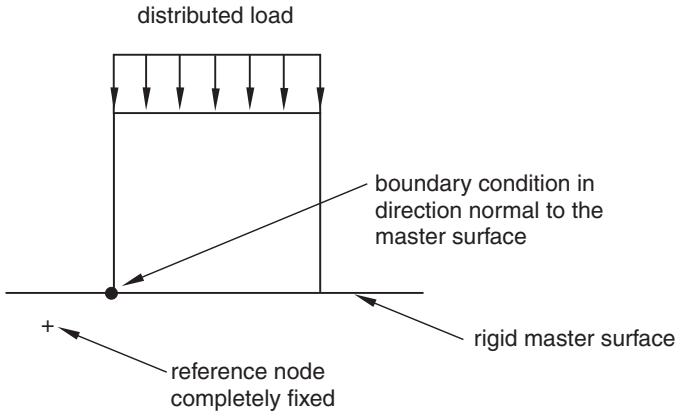


Figure 33.6.1–10 Overconstraints involving normal contact interactions and boundary conditions.

The contact and boundary conditions related to overconstraints are removed automatically only if the master surface is defined as an analytical rigid surface. In all other cases, if an overconstraint occurs during the analysis, a zero pivot message is issued by the equation solver (see below) and the chains of constraints responsible for the overconstraint are clearly outlined.

Checks related to Lagrange friction

A common redundant constraint case is depicted in Figure 33.6.1–11. The symmetry boundary conditions combined with the Lagrange friction are redundant. The slave node is in contact and the tangent to the surface is in approximately the same direction as the specified boundary condition at the slave node. To avoid redundancy, at this node Abaqus/Standard will switch from the Lagrange friction formulation to the default penalty formulation ("Frictional behavior," Section 35.1.5) if the motion of the master nodes is prescribed in the tangent direction.

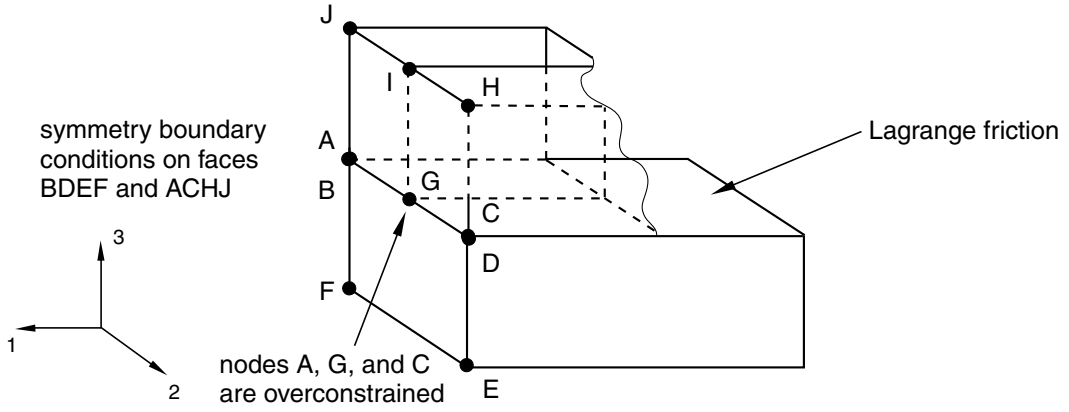


Figure 33.6.1–11 Lagrange friction and boundary conditions.

Overconstraints detected in the equation solver

All overconstraints that cannot be identified and resolved during preprocessing or during the analysis need to be detected by the equation solver. Examples include models with contact interactions where slave nodes are driven by specified boundary conditions into partially fixed rigid surfaces; contact with multiple master surfaces; closed-loop and multiple-loop mechanisms in which rigid bodies are connected by connector elements; and many more. By default, equation solver overconstraint checks are performed continuously during the analysis.

Abaqus/Standard will not resolve overconstraints detected by the equation solver. Instead, detailed messages with information regarding the kinematic constraints involved in the overconstraint will be issued. The message first identifies the nodes involved in either a consistent or an inconsistent overconstraint by using zero pivot information from the Gauss elimination in the solver ("Direct linear equation solver," Section 6.1.4). A detailed message containing constraint information is then issued.

The 4-bar mechanism shown in Figure 33.6.1–12 illustrates this strategy. Four three-dimensional rigid bodies are defined as follows: the rigid body with reference node 10001 includes nodes 2 and 101; the rigid body with reference node 10002 includes nodes 3 and 102; the rigid body with reference node 10003 includes nodes 4 and 103; and the rigid body with reference node 10004 includes nodes 1 and 104. The four rigid bodies are connected with four JOIN and REVOLUTE combination connector elements defined as follows: element 20001 between nodes 1 and 101; element 20002 between nodes 2 and 102; element 20003 between nodes 3 and 103; and element 20004 between nodes 4 and 104. Each connector element enforces three translation and two rotation constraints ("Connectors: overview," Section 30.1.1), and all four revolute axis directions are parallel. The bottom rigid body (with reference node 10004) is fixed. The motion of the bottom left REVOLUTE connector (element 20001) is prescribed to rotate the mechanism.

When Abaqus/Standard attempts to find a solution for this model, three zero pivots are identified in the first increment of the analysis suggesting that there are three constraints too many in the model.

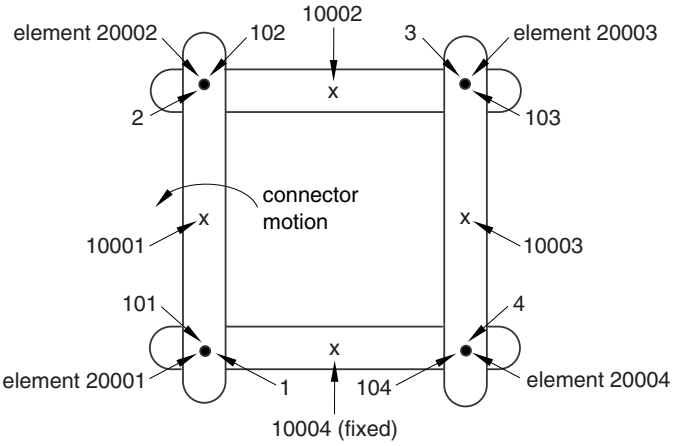


Figure 33.6.1–12 Hard-to-detect redundant constraints.

Eventually, one would have to remove three constraints to render the model properly constrained. In this simple example a count of the degrees of freedom and constraints confirms the number of overconstraints, as follows. There are four rigid bodies in the model, with a total of 24 degrees of freedom. The reference node 10004 is completely fixed with a boundary condition, constraining six degrees of freedom; and the prescribed connector motion enforces one rotational constraint, constraining one degree of freedom. Hence, there are 17 degrees of freedom remaining. Each of the four connector elements enforces five constraints, for a total of 20 constraints. Thus, there are three constraints too many in the model, which matches the number of zero pivots identified by the equation solver. To help you identify the constraints that should be removed, the following message is produced in the message (**.msg**) file outlining the chains of constraints that generated the overconstraint:

```
 ***WARNING: SOLVER PROBLEM. ZERO PIVOT WHEN PROCESSING ELEMENT 20004
             INTERNAL NODE 1 D.O.F. 4

OVERCONSTRAINT CHECKS:  An overconstraint was detected at one of the
Lagrange multipliers associated with element 20004. There are
multiple constraints applied directly or chained constraints that
are applied indirectly to this element. The following is a list of
nodes and chained constraints between these nodes that most likely
lead to the detected overconstraint.

LAGRANGE MULTIPLIER: 4 <-> 104: connector element 20004 type
                     JOIN REVOLUTE constraining 3 translations
                     and  2 rotations
..4 -> 10003: *RIGID BODY (or *COUPLING-KINEMATIC)
....10003 -> 103: *RIGID BODY (or *COUPLING-KINEMATIC)
```

```
......103 -> 3: connector element 20003 type JOIN REVOLUTE
                 constraining 3 translations  and  2 rotations
........3 -> 10002: *RIGID BODY (or *COUPLING-KINEMATIC)
..........10002 -> 102: *RIGID BODY (or *COUPLING-KINEMATIC)
............102 -> 2: connector element 20002 type JOIN REVOLUTE
                      constraining 3 translations and 2 rotations
..............2 -> 10001: *RIGID BODY (or *COUPLING-KINEMATIC)
................10001 -> 101: *RIGID BODY (or *COUPLING-KINEMATIC)
..................101 -> 1: connector element 20001 type
                            JOIN REVOLUTE constraining  3
                            translations  and 2 rotations
....................1 -> 10004: *RIGID BODY (or *COUPLING-KINEMATIC)
......................10004 -> *BOUNDARY in degrees of freedom
                                 1  2  3  4  5  6
......................10004 -> 104: *RIGID BODY
                                      (or *COUPLING-KINEMATIC)
....................1 -> 101: connector element 20001 with
                               *CONNECTOR MOTION in components  4

Please analyze these constraint loops and remove unnecessary
 constraints.
```

First, the message identifies the user-defined or, in this case, the internally defined (Lagrange multiplier) node at which a zero pivot was identified. A typical line in this output issues information related to one constraint. For example, the first line in this output

```
LAGRANGE MULTIPLIER: 4 <-> 104: connector element 20004 type
                     JOIN REVOLUTE constraining 3 translations
                     and  2 rotations
```

informs you that the Lagrange multiplier on which the zero pivot occurs enforces one of the five constraints (JOIN and REVOLUTE) associated with connector element 20004 between user-defined nodes 4 and 104. Each of the subsequent lines conveys information related to one constraint in the chains of constraints originating at the zero pivot node or in chains adjacent to them. For example, the line

```
....10003 -> 103: *RIGID BODY (or *COUPLING - KINEMATIC)
```

informs you that there is a rigid body constraint between nodes 10003 and 103, while the line

```
......................10004 -> *BOUNDARY in degrees of freedom
                                  1  2  3  4  5  6
```

states that there is a boundary condition constraint fixing degrees of freedom 1 through 6 at node 10004.

Indentation levels (the dots in front of the node numbers) identify links in a chain of constraints. Each time a constraint is found to link another node in a particular chain, the indentation is increased by two dots and the constraint information is printed out. For example, starting from the top of the

message, the Lagrange multiplier is connected to node 4, node 4 is connected to node 10003, node 10003 is connected to node 103, and so on. When the indentation on a certain line is less than or equal to the indentation on the previous line, a chain of constraints has ended on the previous line. For example, a chain has ended on the line

```
.....................10004 -> *BOUNDARY in degrees of freedom
                                  1  2  3  4  5  6
```

since the next line has equal indentation.

Three chains of constraints (in correspondence with the three zero pivots that were found) that most likely generated the overconstraint can be identified in the model above. Starting from the top, one can first identify a chain of constraints that terminates in a boundary condition (ground):

```
Lagrange multiplier: 4 -> 10003 -> 103 -> 3 -> 10002 -> 2 ->
10001 -> 101 -> 1 -> 10004 -> *BOUNDARY
```

Since the indentation of the two lines starting with node 10004 is the same, one should expect another chain of constraints to include the constraint output on the second of the two lines. Indeed, one can identify a closed loop of constraints:

```
Lagrange multiplier : 4-> 10003 -> 103 -> 3 -> 10002 -> 2 ->
10001 -> 101 -> 1 -> 10004 -> 104 <-> 4
```

Finally, since the two lines starting with node 1 have the same indentation, one expects that a separate chain of constraints will include the last line in the output. A third (closed) loop

```
101 -> 1 -> 101
```

is identified.

If the chains of constraints terminate in a free end (not ending in a constraint), the chain does not have any contribution in generating the overconstraint. There are no such chains in this example.

Correcting an overconstrained model

A node set containing all the nodes in the chains of constraints associated with a particular zero pivot is generated automatically and can be displayed in the Visualization module of Abaqus/CAE.

There is no unique way to remove the overconstraints in this model. For example, if one JOIN and REVOLUTE (five constraints) combination is replaced with a SLOT connector element, which enforces only the two translation constraints in the plane of the mechanism, there are no redundancies. Alternatively, you could remove the REVOLUTE from one of the connector elements and also use a SLOT connection instead of a JOIN in one of the other connector elements.

Another alternative is to relax some of the constraints. In the example outlined here, an elastic body could replace one or more of the rigid bodies. You could also relax the Lagrange multiplier-based constraints (e.g., JOIN or REVOLUTE) by using CARTESIAN and CARDAN connection types with appropriate elastic stiffnesses (see “Connector behavior,” Section 30.2.1).

After analyzing the chains of constraints, you have to decide which constraints have to be removed to render the model properly constrained and also best fit the modeling goals. For this example the

three constraints cannot be removed randomly. Removing any three combinations of the six boundary conditions, for example, would make the problem worse: the model is still overconstrained, and three rigid body modes have been added to the model. Moreover, you should remove the constraints that do not affect the kinematics of the model. For example, you cannot completely remove a JOIN connection from any of the connector elements since the model would be different from that originally intended.

Controlling the overconstraint checks

By default, Abaqus/Standard will attempt to remove as many redundant constraints as possible, as discussed in the sections above. When it is not possible to remove a redundant constraint or an inconsistent overconstraint is detected, a detailed message is issued identifying the constraints contributing to the overconstraint. You can modify this default behavior by prescribing constraint controls for the model or the step.

Overconstraints may produce damaging and unpredictable behavior. Therefore, it is strongly recommended that overconstraint checking be used in both the preprocessor and during the analysis at least during the first running of a model. Furthermore, it is recommended that the original model be changed to correct any overconstraints identified by Abaqus/Standard. Only after establishing confidence that the model is free of overconstraints should constraint checks be turned off. The only advantage of turning off the constraint checks is a minor speedup of the analysis.

Bypassing the overconstraint checks

The overconstraint checks performed during input file preprocessing and during the analysis can be bypassed. Bypassing these checks is not recommended, as it may allow a model with overconstraints to enter into the analysis code. Bypassing the overconstraint checks is not step dependent; i.e., the setting is defined as model data and affects the entire analysis.

Input File Usage: *CONSTRAINT CONTROLS, NO CHECKS

Preventing automatic redundant constraint resolution

Automatic model modifications in the model preprocessor can be prevented. In this case Abaqus/Standard will still perform overconstraint checks, but no automatic redundant constraint resolution will be performed; only appropriate error messages will be issued. Preventing constraint resolution is not step dependent; i.e., the setting is defined as model data and affects the entire analysis.

Input File Usage: *CONSTRAINT CONTROLS, NO CHANGES

Changing the frequency of the overconstraint checks

By default, the overconstraint checks are performed at every increment during the analysis. You can modify the frequency of these checks (in increments) for each step in the analysis. If the frequency is set equal to zero, no overconstraint checks are performed during that analysis step. The frequency specification is maintained in subsequent steps until the value is reset.

Input File Usage: *CONSTRAINT CONTROLS, CHECK FREQUENCY=*n*

Stopping the analysis when overconstraints are detected

By default, the analysis continues even though an overconstraint is detected. This behavior can be changed on a step-dependent basis. The analysis can be stopped the first time an overconstraint is detected in a step, or it can be stopped only if a converged solution is obtained despite the fact that overconstraints exist. This setting is maintained in subsequent steps until it is reset.

Input File Usage: Use one of the following options:

```
*CONSTRAINT CONTROLS, TERMINATE ANALYSIS=FIRST
OCCURRENCE
*CONSTRAINT CONTROLS, TERMINATE ANALYSIS=CONVERGED
```

Part IX: Interactions

- Chapter 34, “Defining Contact Interactions”
- Chapter 35, “Contact Property Models”
- Chapter 36, “Contact Formulations and Numerical Methods”
- Chapter 37, “Contact Difficulties and Diagnostics”
- Chapter 38, “Contact Elements in Abaqus/Standard”
- Chapter 39, “Defining Cavity Radiation in Abaqus/Standard”

34. Defining Contact Interactions

Overview 34.1
Defining general contact in Abaqus/Standard 34.2
Defining contact pairs in Abaqus/Standard 34.3
Defining general contact in Abaqus/Explicit 34.4
Defining contact pairs in Abaqus/Explicit 34.5

34.1 Overview

- “Contact interaction analysis: overview,” Section 34.1.1

34.1.1 CONTACT INTERACTION ANALYSIS: OVERVIEW

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

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 (i.e., 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 "Surfaces: overview," Section 2.3.1). Abaqus has four classifications of contact surfaces:

- element-based deformable and rigid surfaces ("Element-based surface definition," Section 2.3.2);
- node-based deformable and rigid surfaces ("Node-based surface definition," Section 2.3.3);
- analytical rigid surfaces ("Analytical rigid surface definition," Section 2.3.4); and
- Eulerian material surfaces for Abaqus/Explicit ("Eulerian surface definition," Section 2.3.5).

Surfaces of the same type can be combined to create new surfaces (see "Operating on surfaces," Section 2.3.6). 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 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 "Defining contact pairs in Abaqus/Standard," Section 34.3.1, and "Defining contact pairs in Abaqus/Explicit," Section 34.5.1. The definition of general contact interactions is discussed in detail in "Defining general contact interactions in Abaqus/Standard," Section 34.2.1, and "Defining general contact interactions in Abaqus/Explicit," Section 34.4.1.

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," Section 34.3.2, and "Assigning surface properties for contact pairs in Abaqus/Explicit," Section 34.5.2. Surface properties for general contact are discussed in "Surface properties for general contact in Abaqus/Standard," Section 34.2.2, and "Assigning surface properties for general contact in Abaqus/Explicit," Section 34.4.2.

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 "Mechanical contact properties: overview," Section 35.1.1). Some of the available models are:

- softened contact ("Contact pressure-overclosure relationships," Section 35.1.2, and "Frictional behavior," Section 35.1.5);
- contact damping ("Contact damping," Section 35.1.3, and "Frictional behavior," Section 35.1.5);
- friction ("Frictional behavior," Section 35.1.5);
- a user-defined constitutive model for surface interactions ("User-defined interfacial constitutive behavior," Section 35.1.6); and
- spot welds bonding two surfaces together until the welds fail ("Breakable bonds," Section 35.1.9).

The thermal, thermal-electrical, and pore-fluid surface interaction models available in Abaqus are discussed in "Thermal contact properties," Section 35.2.1; "Electrical contact properties," Section 35.3.1; and "Pore fluid contact properties," Section 35.4.1, 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," Section 34.3.3, and "Assigning contact properties for contact pairs in Abaqus/Explicit," Section 34.5.3. Information on assigning contact properties to general contact interactions can be found in "Contact properties for general contact in Abaqus/Standard," Section 34.2.3, and "Assigning contact properties for general contact in Abaqus/Explicit," Section 34.4.3.

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 master and slave 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," Section 36.1.1, and "Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2. The available numerical controls for the various contact algorithms are discussed in "Numerical controls for general contact in Abaqus/Standard," Section 34.2.6; "Adjusting contact controls in Abaqus/Standard," Section 34.3.6; "Contact controls for general contact in Abaqus/Explicit," Section 34.4.5; and "Contact controls for contact pairs in Abaqus/Explicit," Section 34.5.5.

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 the 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 thermal-mechanical 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 thermal-mechanical 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 will often result in more efficient or robust 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," Section 15.6 of the Abaqus/CAE User's Manual).

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: General contact uses the finite-sliding, surface-to-surface formulation. 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," Section 36.1.1, 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. 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," Section 34.2.2, and "Assigning surface properties for contact pairs in Abaqus/Standard," Section 34.3.2, 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," Section 36.1.2, 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 "Controlling initial contact status in Abaqus/Standard," Section 34.2.4; "Modeling contact interference fits in Abaqus/Standard," Section 34.3.4; and "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5; for more information on contact initialization in Abaqus/Standard.
- Master-slave assignments: General contact automatically assigns pure master and slave roles for most contact interactions and automatically assigns balanced master-slave roles to other contact interactions. The user must assign master and slave roles for most contact pairs. See "Numerical controls for general contact in Abaqus/Standard," Section 34.2.6, and "Choosing the master and slave roles in a two-surface contact pair" in "Contact formulations in Abaqus/Standard," Section 36.1.1, for discussions of master and slave roles for contacting surfaces.

The first three differences listed above disappear if you specify the finite-sliding, surface-to-surface formulation for contact pairs.

Additional contact pair capabilities

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

- Contact involving analytical rigid surfaces or rigid surfaces defined with user subroutine **RSURFU** (however, element-based rigid surfaces can be included in either general contact or contact pairs).
- Contact involving node-based surfaces or surfaces on three-dimensional beam elements.
- Small-sliding contact and tied contact.
- The finite-sliding, node-to-surface contact formulation.
- Debonding and cohesive contact behavior.
- Surface interactions in analyses without displacement degrees of freedom, such as pure heat transfer.
- Pressure-penetration loading.
- Local definitions of some numerical contact controls.
- Symmetric model generation.

A single analysis can include general contact and contact pair definitions. For example, you may choose to model contact interactions involving analytical rigid surfaces with contact pairs and other contact interactions with general contact. General contact automatically avoids processing contact interactions that are treated by contact pairs.

Contact simulations requiring contact elements

Surface-based contact methods associated with general contact and contact pairs cannot be used for certain classes of problems. Abaqus/Standard provides a library of contact elements for these 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," Section 38.3.1).
- 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," Section 38.2.1).
- Simulations using axisymmetric elements with asymmetric deformations, CAXA*n* and SAXA*n* elements. See "Contact modeling if asymmetric-axisymmetric elements are present," Section 34.3.10, 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," Section 38.2.1).

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 Chapter 38, "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. The general ("automatic") contact algorithm allows very simple definitions of contact with very few restrictions on the types of surfaces involved (see "Defining general contact in Abaqus/Explicit," Section 34.4). The contact pair algorithm has more restrictions on the types of surfaces involved and often requires more careful definition of contact; however, it allows for some interaction behaviors that currently are not available with the general contact algorithm (see "Defining contact pairs in Abaqus/Explicit," Section 34.5). The general contact and contact pairs algoirthms 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.
- 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,” Section 2.3.5).

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

- 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” in “Eulerian analysis,” Section 14.1.1).
- 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 slave node can see contact with multiple facets per increment; with the contact pair algorithm each slave 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 does not place any restrictions on the domain decomposition for domain level parallelization (see “Parallel execution in Abaqus/Explicit,” Section 3.5.3).

- The general contact algorithm in Abaqus/Explicit has been developed to minimize the need for algorithmic controls.

See "Knee bolster impact with general contact," Section 2.1.9 of the Abaqus Example Problems Manual; "Crimp forming with general contact," Section 2.1.10 of the Abaqus Example Problems Manual; and "Collapse of a stack of blocks with general contact," Section 2.1.11 of the Abaqus Example Problems Manual, 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:

- Two-dimensional surfaces
- Kinematically enforced contact (see "Contact constraint enforcement methods in Abaqus/Explicit," Section 36.2.3; the general contact algorithm uses only penalty enforcement)
- Small-sliding contact (see "Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2)
- Exponential and no separation contact pressure-overclosure models
- Breakable bonds, such as spot welds (however, mesh-independent spot welds can be used with either contact algorithm; see "Mesh-independent fasteners," Section 33.3.4)

In addition, the general contact algorithm in Abaqus/Explicit places more restrictions on adaptive meshing than the contact pair algorithm (see "Defining ALE adaptive mesh domains in Abaqus/Explicit," Section 12.2.2). The choice of contact algorithm may affect the speedup factor if loop-level parallelization is used: the contact pair algorithm includes some loop-level parallelization, while the general contact algorithm has no loop-level parallelization. Contact output is more complete for a contact pair analysis.

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 "Defining contact pairs in Abaqus/Standard," Section 34.3.1). In the contact pair algorithm in Abaqus/Explicit contact constraints are history definition data (see "Defining a model in Abaqus," Section 1.3.1); in the general contact algorithm in Abaqus/Explicit contact definitions can be either model or history data.
- Abaqus/Standard typically uses a pure master-slave relationship for the contact constraints; whereas Abaqus/Explicit typically uses balanced master-slave 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 formulations, which Abaqus/Explicit does not;
  - Abaqus/Explicit provides an edge-to-edge formulation, which Abaqus/Standard 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 master nodes according to the current position of the slave 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,” Section 9.2.2). However, in many cases you can successfully respecify a contact definition in an import analysis.

34.2 Defining general contact in Abaqus/Standard

- "Defining general contact interactions in Abaqus/Standard," Section 34.2.1
- "Surface properties for general contact in Abaqus/Standard," Section 34.2.2
- "Contact properties for general contact in Abaqus/Standard," Section 34.2.3
- "Controlling initial contact status in Abaqus/Standard," Section 34.2.4
- "Stabilization for general contact in Abaqus/Standard," Section 34.2.5
- "Numerical controls for general contact in Abaqus/Standard," Section 34.2.6

34.2.1 DEFINING GENERAL CONTACT INTERACTIONS IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Contact interaction analysis: overview," Section 34.1.1
- *CONTACT
- *CONTACT INCLUSIONS
- *CONTACT EXCLUSIONS
- "Defining general contact," Section 15.13.1 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Abaqus/Standard provides two algorithms for modeling contact and interaction problems: the general contact algorithm and the contact pair algorithm. See "Contact interaction analysis: overview," Section 34.1.1, 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 that may be 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 (i.e., 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
- uses the finite-sliding, surface-to-surface contact formulation.

Defining a general contact interaction

The definition of a general contact interaction consists of specifying:

- the general contact algorithm and defining the contact domain (i.e., the surfaces that interact with one another), as described in this section;
- the contact surface properties ("Surface properties for general contact in Abaqus/Standard," Section 34.2.2);
- the mechanical contact property models ("Contact properties for general contact in Abaqus/Standard," Section 34.2.3);

- the controls associated with the initial contact state ("Controlling initial contact status in Abaqus/Standard," Section 34.2.4); and
- the algorithmic contact controls ("Numerical controls for general contact in Abaqus/Standard," Section 34.2.6).

An example of an analysis that uses general contact to define contact between the various components of an assembly is described in "Impact analysis of a pawl-ratchet device," Section 2.1.17 of the Abaqus Example Problems Manual.

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 "Contact interaction analysis: overview," Section 34.1.1. For detailed information on defining surfaces in Abaqus/Standard for use with the general contact algorithm, see "Element-based surface definition," Section 2.3.2.

A convenient method of specifying the contact domain is using cropped surfaces. Such surfaces can be used to perform "contact in a box" by using a contact domain that is enclosed in a specified rectangular box in the original configuration. For more information, see "Operating on surfaces," Section 2.3.6.

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.

The general contact algorithm in Abaqus/Standard uses the surface-to-surface contact formulation as the primary formulation and can use the edge-to-surface contact formulation as a supplementary formulation. The general contact algorithm does not consider contact involving analytical surfaces or node-based surfaces, although these surface types can be included in contact pairs in analyses that also use general contact.

Considerations for edge-to-surface contact

The general contact algorithm can consider three-dimensional edge-to-surface contact, which is more effective at resolving some interactions than the surface-to-surface contact formulation. The edge-to-surface contact formulation is primarily intended to avoid localized penetration of a feature's edge of one surface into a relatively smooth portion of another surface when the normal directions of the respective surface facets in the active contact region form an oblique angle. The model shown in Figure 34.2.1–1 will benefit from supplementary edge-to-surface contact enforcement because the active contact zone corresponds to a feature edge during some periods of the insertion loading. Supplementary edge-to-surface contact enforcement is not necessary for the model shown in Figure 34.2.1–2 because the surface-to-surface contact formulation is able to adequately resist the penetrations.

By default, when a surface is used in a general contact interaction, all applicable facets are included in the contact definition; however, you specify which, if any, edges to consider for edge-to-surface contact (see "Feature edges" in "Surface properties for general contact in Abaqus/Standard," Section 34.2.2). Edge-to-surface contact constraints never participate in thermal, electrical, or pore pressure contact properties. For example, in a coupled temperature-displacement analysis,

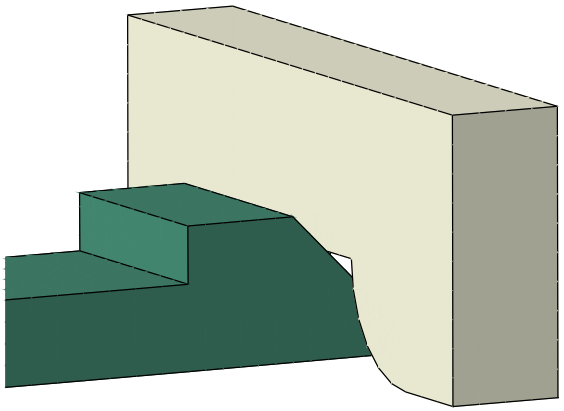


Figure 34.2.1–1 Snap-fit example involving feature edge-to-surface contact with an oblique angle between surface normals in the contact region.

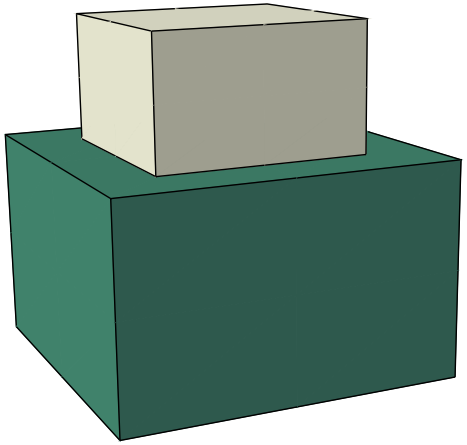


Figure 34.2.1–2 Example with feature-edges at the perimeter of an active contact region that has opposing surface normals.

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.

Both surface-to-surface and edge-to-surface contact constraints may be active at the same nodes. To help avoid numerical overconstraint issues, edge-to-surface contact constraints are always enforced with a penalty method.

Including general contact in an analysis

General contact in Abaqus/Standard is defined at the beginning of an analysis. Only one general contact definition can be specified, and this definition is in effect for every step of the analysis.

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

*CONTACT

This option can appear only once in the model definition.

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

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. Only one contact inclusions definition and one contact exclusions definition are allowed in the model definition.

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/Standard. 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/Standard. This default surface contains, with the exceptions noted below, all exterior element faces. 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,” Section 31.5.3, 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; any data lines specified will be ignored.

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 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. In some cases computational performance and robustness can be improved by including only portions of surfaces in the general contact domain that will experience contact during an analysis.

Multiple surface pairings can be included in the contact domain. All of the surfaces specified must be element-based surfaces.

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 (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 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. All of 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*

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 (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 it is not possible for a rigid body to contact itself.
- When you specify pure master-slave contact surface weighting for a particular general contact surface pair, contact exclusions are generated automatically for the master-slave orientation opposite to that specified (see "Numerical controls for general contact in Abaqus/Standard," Section 34.2.6, for more information on this type of contact exclusion).
- Abaqus/Standard assigns default pure master-slave roles for contact involving disconnected bodies within the general contact domain, and contact exclusions are generated by default for the opposite master-slave orientations. Options to override the default pure master-slave assignments with alternative pure master-slave assignments or balanced master-slave assignments are discussed in "Numerical controls for general contact in Abaqus/Standard," Section 34.2.6.
- Contact exclusions are generated automatically for portions of surfaces that are severely overclosed in the initial configuration of the model. See "Controlling initial contact status in Abaqus/Standard," Section 34.2.4, for more information.

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
```

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," Section 37.1.1, 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 Chapter 35, "Contact Property Models."

General contact domain surfaces

Abaqus/Standard generates the following internal surfaces when a general contact domain is defined: **`General_Contact_Faces`**, **`General_Contact_Edges`**, and **`General_Contact_Component_`*`k`***, where ***`k`*** corresponds to the component number. The automatically generated "component" surfaces are related to the default assignment of master-slave roles for contact between disconnected surface regions: the lowered-number component surfaces will act as master surfaces to the higher-numbered component surfaces. Component surfaces are also referred to in diagnostic messages associated with general contact. **`General_Contact_Faces`** contains all surface faces that are included in the general contact domain. **`General_Contact_Edges`** contains all surface edges that satisfy the criteria to participate in edge-to-surface contact. These internal surfaces can be viewed using display groups in the Visualization module of Abaqus/CAE. The 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 slave or master nodes. In such cases the nodal values are generally affected by more than one contact constraint. Other nodal contact variables are available only at nodes acting as slave nodes. In these cases the value at each slave node reflects a value associated with a particular contact constraint. Most contact output to the data (**`.dat`**) and results (**`.fil`**) files is associated with individual constraints.

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," Section 12.3.2, 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.

- To minimize contact stress noise within a region of active contact, Abaqus/Standard computes nodal contact stresses as weighted averages of values associated with active contact constraints in which a node participates. Some filtering is applied to reduce the contact stress values reported for nodes on the fringe of the active contact region (that only weakly participate in contact constraints), but this filtering is not "perfect," which can result in the contact zone size appearing somewhat exaggerated. Similarly, contact status output will also be 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 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 slave nodes. For example, CFN is the total force acting on slave 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 master and slave 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," Section 4.2.1.

Output requests can be limited by specifying a node set containing a subset of the nodes acting as slave 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" in "Output to the data and results files," Section 4.1.2.
- To request output to the output database (`.odb`) file, see "Surface output in Abaqus/Standard and Abaqus/Explicit" in "Output to the output database," Section 4.1.3.

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 slip directions defined on the surfaces. The definition of slip directions is explained in "Local tangent directions on a surface" in "Contact formulations in

Abaqus/Standard," Section 36.1.1. 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 slip 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 slip direction. For example, CSHEAR1 is the frictional shear stress component in the first slip direction, while CSHEAR2 is the frictional shear stress component in the second slip 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 slip direction. The incremental relative nodal displacement vector measures the motion of a slave node relative to the motion of the master surface. The incremental slip is accumulated only when the slave node is contacting the master 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," Section 5.1.1 of the Abaqus Theory Manual; "Finite-sliding interaction between deformable bodies," Section 5.1.2 of the Abaqus Theory Manual; and "Finite-sliding interaction between a deformable and a rigid body," Section 5.1.3 of the Abaqus Theory Manual.

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

34.2.2 SURFACE PROPERTIES FOR GENERAL CONTACT IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Defining general contact interactions in Abaqus/Standard," Section 34.2.1
- *CONTACT
- *SURFACE PROPERTY ASSIGNMENT
- "Specifying surface property assignments for general contact," Section 15.13.4 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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;
- can be applied selectively to particular regions within a general contact domain; and
- cannot be applied to analytical rigid surfaces.

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.

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**

Surface geometry correction

By default, contact calculations are based on unsmoothed, faceted representations of the finite element surfaces in a general contact domain. An optional contact smoothing technique simulates a more realistic representation of curved surfaces in the contact calculations, resulting in improved contact stress and pressure accuracy. This contact smoothing technique is discussed in "Smoothing contact surfaces in Abaqus/Standard," Section 36.1.3.

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, ORIGINAL (default)

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

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Surface thickness assignments: Edit**:
Select surface, click the arrows to transfer surface 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, *value*

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

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Surface thickness assignments: Edit**:
Select surface, click the arrows to transfer surface 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 "Controlling initial contact status in Abaqus/Standard," Section 34.2.4) 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, *value or label*, *scale_factor*

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

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Surface thickness assignments: Edit**:
Select surface, click the arrows to transfer surface 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, ORIGINAL

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

*SURFACE PROPERTY ASSIGNMENT, PROPERTY=OFFSET FRACTION
surface, *offset*

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.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Standard)**: **Surface Properties**: **Shell/Membrane offset assignments: Edit**: Select surface, and click the arrows to transfer surface 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

General contact in Abaqus/Standard includes a supplementary edge-to-surface contact formulation for feature edges of solid bodies, as discussed in "Defining general contact interactions in Abaqus/Standard," Section 34.2.1; however, you must specify a feature angle criterion for any feature edges to be considered.

The feature angle is the angle formed between the normals of the two facets connected to an edge. The angles between facets are based on the initial configuration. A negative angle will result at concave meetings of facets. Figure 34.4.2–4 shows some examples of how the feature angle is calculated for different edges.

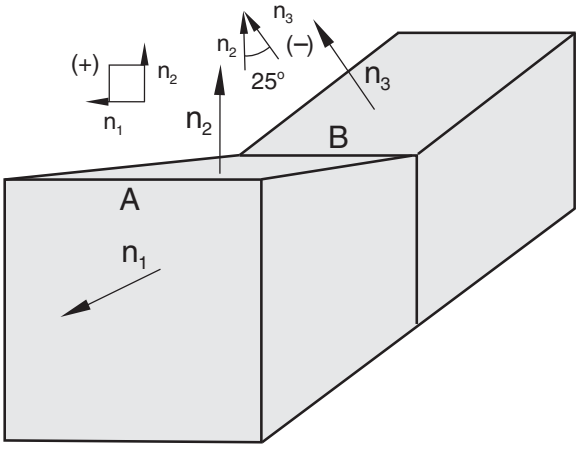


Figure 34.2.2–1 Calculating the feature angle.

The feature angle for edge A is 90° (the angle between n_1 and n_2); the feature angle for edge B is −25° (the angle between n_2 and n_3).

When you assign a feature angle criterion, geometric edges of solid 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 "Defining general contact interactions in Abaqus/Standard," Section 34.2.1) apply to both the surface-to-surface contact formulation and the edge-to-surface 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 (90° > 20°) but edge B would not be activated (−25° < 20°). The cutoff feature angle cannot be set to less than 20°. Allowing a cutoff feature angle of less than 20° could severely degrade performance and would not affect the analysis results significantly compared to a cutoff angle of 20°.

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," Section 34.2.3, for further discussion of contact properties for general contact). The edge-to-surface formulation always uses the penalty enforcement method and only involves displacement degrees of freedom. For example, the edge-to-surface formulation does not contribute to thermal gap conductance across a contact interface.

Input File Usage: *SURFACE PROPERTY ASSIGNMENT, PROPERTY=FEATURE
EDGE CRITERIA
surface, *feature_angle_value*

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

Abaqus/CAE Usage: A feature angle criterion cannot be defined in Abaqus/CAE.

34.2.3 CONTACT PROPERTIES FOR GENERAL CONTACT IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Defining general contact interactions in Abaqus/Standard," Section 34.2.1
- "Mechanical contact properties: overview," Section 35.1.1
- "Contact pressure-overclosure relationships," Section 35.1.2
- "Contact damping," Section 35.1.3
- "Frictional behavior," Section 35.1.5
- ∗CONTACT
- ∗CONTACT PROPERTY ASSIGNMENT
- ∗SURFACE INTERACTION
- "Specifying and modifying contact property assignments for general contact," Section 15.13.2 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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.

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, *surface_2*, *interaction_property_name*

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.

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. 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 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 surfaces 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" in "Frictional behavior," Section 35.1.5.

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
```

34.2.4 CONTROLLING INITIAL CONTACT STATUS IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Defining general contact interactions in Abaqus/Standard," Section 34.2.1
- *CONTACT INITIALIZATION ASSIGNMENT
- *CONTACT INITIALIZATION DATA
- "Creating contact initializations," Section 15.12.4 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Contact initialization controls for general contact in Abaqus/Standard:

- can be used to specify whether initial overclosures should be resolved without generating stresses and strains or treated as interference fits that are gradually resolved over multiple increments; and
- can be used to 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 34.2.4–1. These adjustments are intended to correct only minor mismatches associated with mesh generation.

General contact automatically assigns master and slave roles for contact interactions, as discussed in "Numerical controls for general contact in Abaqus/Standard," Section 34.2.6. Abaqus/Standard calculates an overclosure tolerance based on the size of the underlying element facets on a slave surface. Slave surfaces in a particular interaction are repositioned onto the associated master 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 slave 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 master surface. Therefore, general contact does not create interactions between surfaces (or portions of

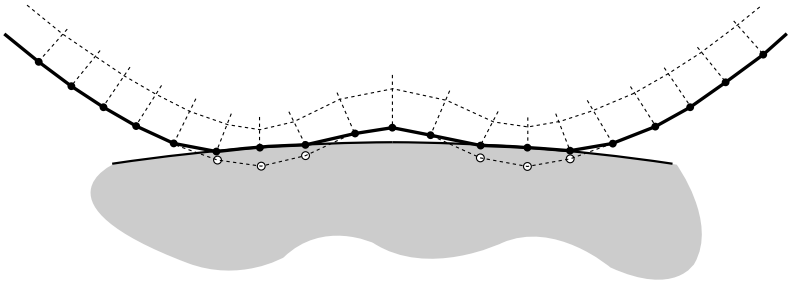


Figure 34.2.4–1 Configuration of contact surfaces after strain-free adjustments to resolve overclosure.

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 finite-sliding, 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 may 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; slave surfaces that lie within these search distances are repositioned directly onto their associated master surface using

strain-free nodal adjustments. Abaqus/Standard takes shell thickness into account when calculating these search distances.

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 may cause mesh distortion when nodes are repositioned over large distances.

Input File Usage: *CONTACT INITIALIZATION DATA, SEARCH ABOVE=*value*, SEARCH BELOW=*value*

Abaqus/CAE Usage: Interaction module: **Interaction→Contact Initialization→Create**: **Resolve with strain-free adjustments**: **Specify opening tolerance:** *value*, **Specify overclosure tolerance:** *value*

Specifying an initial clearance distance

By default, the strain-free adjustments discussed above will 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 will occur only for regions that satisfy the search zone tolerances, as discussed above. Mesh distortion can occur if large strain-free adjustments are necessary to achieve the specified initial clearance distance.

Input File Usage: *CONTACT INITIALIZATION DATA, INITIAL CLEARANCE=*value*

Modeling interference fits

Optionally, the general contact algorithm in Abaqus/Standard can treat initial overclosures as interference fits. The general contact algorithm uses a shrink-fit method to gradually resolve the interference distance over the first step of the analysis (if multiple load increments are used for the first step) as shown in Figure 34.2.4–2, 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.

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 may be observed at slave nodes when surface-to-surface constraints are in a zero-penetration state.

Input File Usage: *CONTACT INITIALIZATION DATA, INTERFERENCE FIT

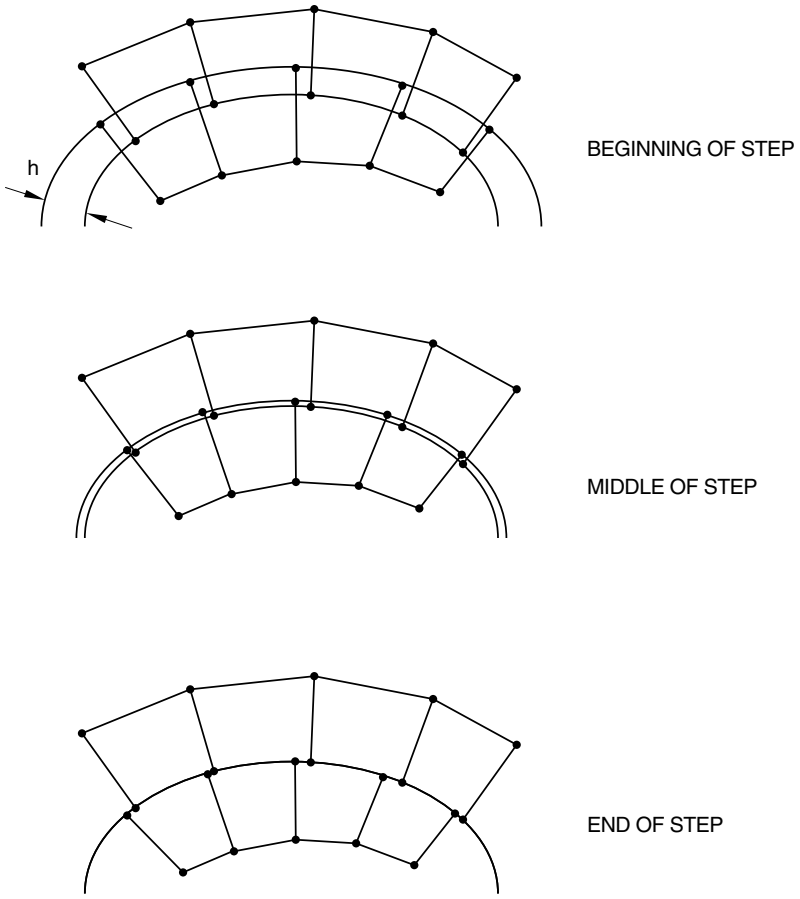


Figure 34.2.4–2 Gradual resolution of contact interference fit.

Abaqus/CAE Usage: Interaction module: **Interaction→Contact Initialization→Create**: **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 slave surface (see "Default contact initialization method" above). An interference fit between two surfaces affects only those slave 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=*value*

Abaqus/CAE Usage: Interaction module: **Interaction→Contact Initialization→Create**: **Resolve penetrations as an interference fit**: **Specify overclosure tolerance:** *value*

Specifying the interference distance

By default, the interference distance is implied by the initial overclosure of the mesh; alternatively, you can specify the interference distance. 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 discuss above, as depicted in Figure 34.2.4–3. Mesh distortion can occur if large strain-free adjustments are necessary to achieve the specified interference distance.

The search region for the strain-free adjustments and subsequent shrink fit resolution is at least at 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=*value*, SEARCH ABOVE=*value*, SEARCH BELOW=*value*

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 first step while interference fits are resolved using the "change friction" method discussed in "Changing friction properties during an Abaqus/Standard analysis" in "Frictional behavior," Section 35.1.5.

Cases in which interference fit resolution with contact pairs is preferred

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 slave 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 master surface normal directions. Figure 34.2.4–4 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 using the contact pair algorithm (see "Modeling contact interference fits in Abaqus/Standard," Section 34.3.4).

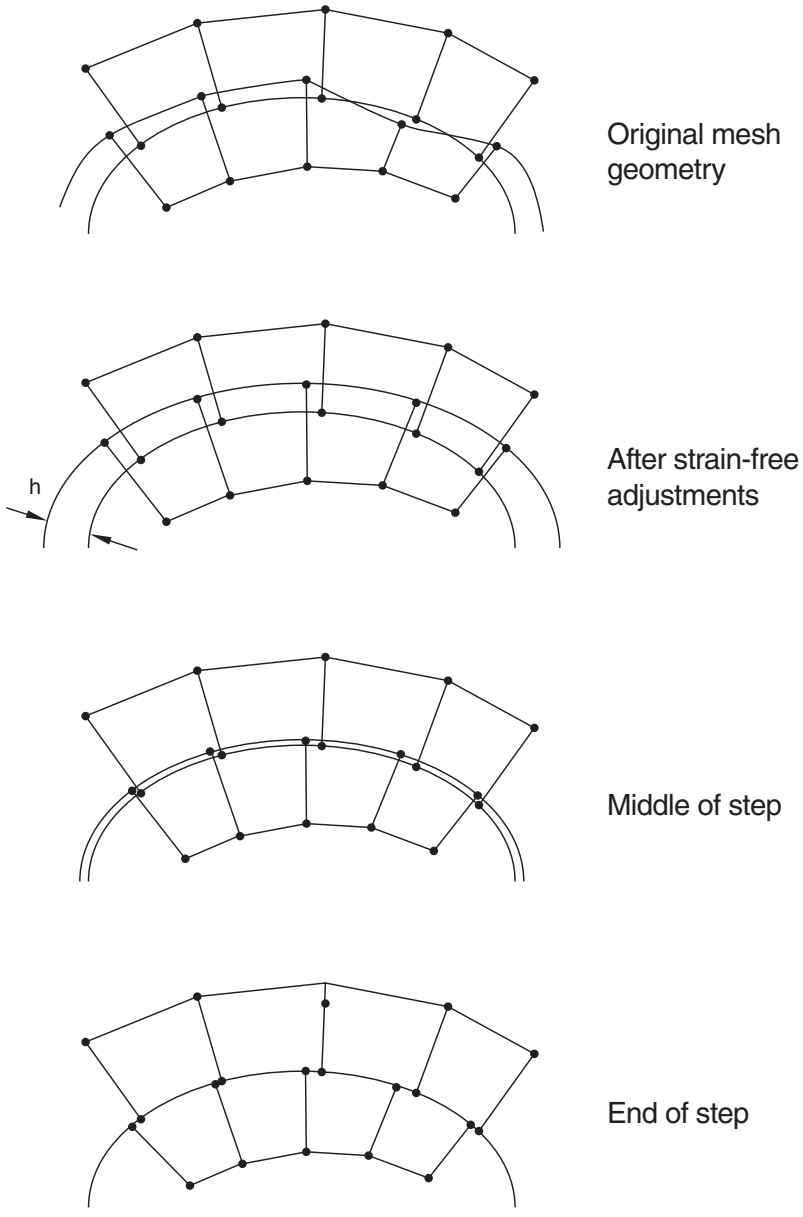


Figure 34.2.4–3 Treatment of a specified interference distance that differs from the interference implied by the original mesh.

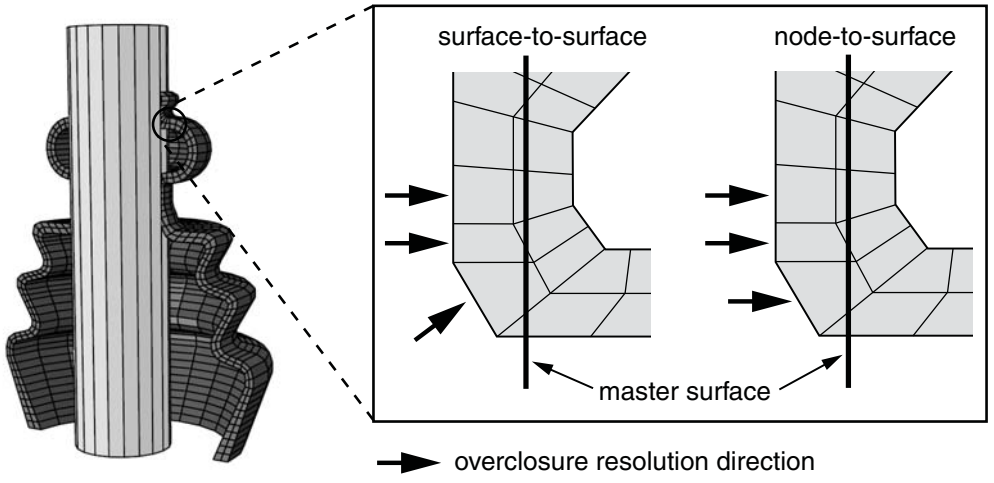


Figure 34.2.4–4 Comparison of contact formulations in an example with a large interference fit.

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 will be 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.

Input File Usage: Use the following option to assign contact inititialization 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 DATA option in the model portion of the input file.

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 34.2.4–5 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 another 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 34.2.4–5 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 data in a general contact analysis:

- 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 bottom left of Figure 34.2.4–5).

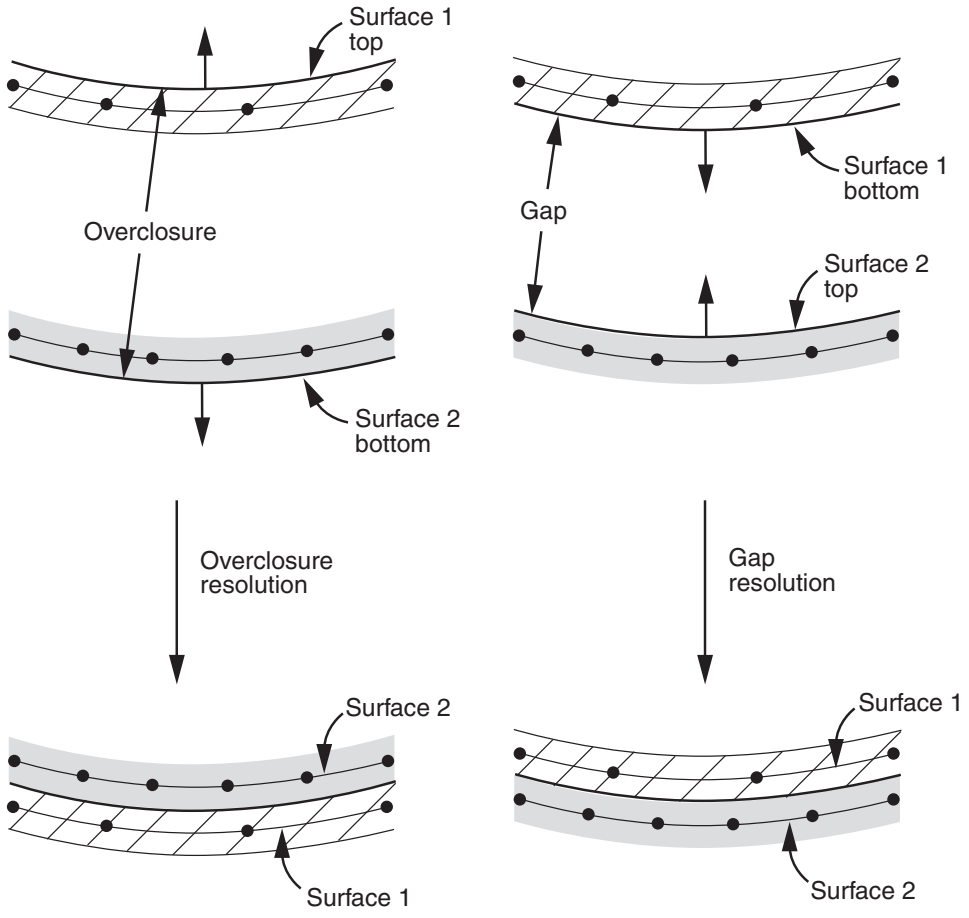


Figure 34.2.4–5 Strain-free adjustments during contact initialization for single-sided shell surfaces.

```
*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
```

34.2.5 STABILIZATION FOR GENERAL CONTACT IN Abaqus/Standard

Product: Abaqus/Standard

References

- "Defining general contact interactions in Abaqus/Standard," Section 34.2.1
- *CONTACT
- *CONTACT STABILIZATION

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 increment relative motion between nearby surfaces, in the same manner as contact damping (see "Contact damping," Section 35.1.3). 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" in "Common difficulties associated with contact modeling in Abaqus/Standard," Section 37.1.2), 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 semi-automated factors to facilitate achieving the desired stabilization characteristics:

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

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

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_{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:

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

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

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.

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.

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}

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}

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

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*

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” in “Adjusting contact controls in Abaqus/Standard,” Section 34.3.6.
- Third priority is given to the default contact stabilization discussed in “Contact at a single point” in “Common difficulties associated with contact modeling in Abaqus/Standard,” Section 37.1.2.

34.2.6 NUMERICAL CONTROLS FOR GENERAL CONTACT IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Defining general contact interactions in Abaqus/Standard," Section 34.2.1
- *CONTACT
- *CONTACT FORMULATION
- *CONTACT CONTROLS
- "Specifying master-slave assignments for general contact," Section 15.13.5 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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 master-slave roles and the sliding formulation; and
- in some cases can be applied selectively to particular regions within a general contact domain.

Contact formulation

The general contact algorithm uses the finite-sliding, surface-to-surface contact formulation, which is discussed in "Contact formulations in Abaqus/Standard," Section 36.1.1. Other contact formulations are not available for general contact in Abaqus/Standard.

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," Section 36.1.2. Assignment of contact properties to general contact interactions is discussed in "Contact properties for general contact in Abaqus/Standard," Section 34.2.3.

Numerical controls for friction

Numerical controls associated with friction are discussed in "Frictional behavior," Section 35.1.5.

Master and slave roles

The surface-to-surface contact formulation used by general contact generates individual contact constraints using a master-slave approach, as discussed in "Contact formulations in Abaqus/Standard," Section 36.1.1. Abaqus/Standard assigns default pure master-slave roles for contact involving disconnected bodies within the general contact domain. Internal surfaces are generated automatically using the naming convention **General_Contact_Component_*k***, where ***k*** corresponds to an automatically assigned component number. By default, the lowered-number component surfaces will act as master surfaces to the higher-numbered component surfaces. You can determine the default pure master-slave roles by viewing the automatically generated internal surfaces in the Visualization module of Abaqus/CAE (see Chapter 78, "Using display groups to display subsets of your model," of the Abaqus/CAE User's Manual). Self-contact within a body is treated with balanced master-slave contact by default, with each surface node acting as a master node in some constraints and as a slave 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_Component_1**
- **General_Contact_Component_2**
- **General_Contact_Component_3**

By default, the first surface listed acts as a master to the other two, and **General_Contact_Component_2** acts as a master to **General_Contact_Component_3**. Self-contact within each of these three surfaces is modeled with balanced master-slave contact by default.

Specifying non-default master-slave roles

You can override the default master-slave roles by specifying pure master-slave roles or by specifying that balanced master-slave contact should be used. The default master-slave treatment works well in most cases. Keep the following points in mind when modifying the master-slave assignments, in addition to other factors discussed in this section:

- Do not use the internally generated component surfaces when assigning alternative master-slave roles (instead, use surface names that you define).
- The master-slave role assignments are part of the model definition and cannot be modified from step to step.
- The guidelines for assigning pure master-slave roles for contact pairs discussed in "Defining contact between two separate surfaces" in "Defining contact pairs in Abaqus/Standard," Section 34.3.1, are also applicable for situations in which you reassign pure master-slave roles for general contact.
- The limitations of balanced (symmetric) master-slave contact pairs discussed in "Using symmetric master-slave contact pairs to improve contact modeling" in "Defining contact pairs in Abaqus/Standard," Section 34.3.1, are also applicable for situations in which you reassign balanced

master-slave contact for general contact. Balanced master-slave 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 slave surface:

∗CONTACT FORMULATION, TYPE=MASTER SLAVE ROLES
surf_1, *surf_2*, SLAVE

Use the following option to indicate that the first surface should be considered the master surface:

∗CONTACT FORMULATION, TYPE=MASTER SLAVE ROLES
surf_1, *surf_2*, MASTER

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 balanced master-slave contact should be used between two surfaces:

∗CONTACT FORMULATION, TYPE=MASTER SLAVE 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**: **Master-slave 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 master-slave assignments.

In the **First Surface Type** column, enter SLAVE to indicate that the first surface should be considered the slave surface, enter MASTER to indicate that the first surface should be considered the master surface, or enter BALANCED to specify that balanced master-slave contact should be used between the two surfaces.

Automatically generated contact exclusions

Abaqus/Standard automatically generates contact exclusions for the master-slave roles opposite to specified pure master-slave 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 master-slave contact with *surf_A* considered to be the slave surface would result in exclusions being generated internally for master faces of *surf_A* contacting slave 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.

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 slave 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 slave 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.

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 “Adjusting contact controls in Abaqus/Standard,” Section 34.3.6.

34.3 Defining contact pairs in Abaqus/Standard

- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- "Assigning surface properties for contact pairs in Abaqus/Standard," Section 34.3.2
- "Assigning contact properties for contact pairs in Abaqus/Standard," Section 34.3.3
- "Modeling contact interference fits in Abaqus/Standard," Section 34.3.4
- "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5
- "Adjusting contact controls in Abaqus/Standard," Section 34.3.6
- "Defining tied contact in Abaqus/Standard," Section 34.3.7
- "Extending master surfaces and slide lines," Section 34.3.8
- "Contact modeling if substructures are present," Section 34.3.9
- "Contact modeling if asymmetric-axisymmetric elements are present," Section 34.3.10

34.3.1 DEFINING CONTACT PAIRS IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Element-based surface definition," Section 2.3.2
- "Node-based surface definition," Section 2.3.3
- "Analytical rigid surface definition," Section 2.3.4
- "Contact interaction analysis: overview," Section 34.1.1
- *CONTACT PAIR
- *SURFACE
- *MODEL CHANGE
- "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining self-contact," Section 15.13.7 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Using contact and constraint detection," Section 15.16 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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; and
- cannot be formed with one two-dimensional surface and one three-dimensional surface.

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 slave surface such that it includes many nodes that remain separated from the master 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," Section 36.1.1. Interaction property definitions are discussed in "Assigning contact properties for contact pairs in Abaqus/Standard," Section 34.3.3.

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 slave and which will be the master. The selection of master and slave surfaces is discussed in detail in "Choosing the master and slave roles in a two-surface contact pair" in "Contact formulations in Abaqus/Standard," Section 36.1.1. 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 master surface.
- If the surfaces are of comparable size, the surface on the stiffer body should act as the master surface.
- If the surfaces are of comparable size and stiffness, the surface with the coarser mesh should act as the master surface.

Defining contact pairs using the finite-sliding, node-to-surface formulation

Abaqus/Standard uses a finite-sliding, node-to-surface formulation by default.

Input File Usage: *CONTACT PAIR, INTERACTION=*interaction_property_name*
slave_surface_name*, *master_surface_name*

You can also specify the contact discretization directly:

*CONTACT PAIR, INTERACTION=*interaction_property_name*,
TYPE=NODE TO SURFACE
slave_surface_name, *master_surface_name*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface-to-surface contact (Standard)**: select the master surface, click **Surface** or **Node Region**, select the slave 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 slave 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," Section 11.4.3).

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
slave_surface_name, *master_surface_name*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface-to-surface contact (Standard)**: select the master surface, click **Surface**, select the slave 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" in "Contact formulations in Abaqus/Standard," Section 36.1.1.

Input File Usage: *CONTACT PAIR, INTERACTION=*interaction_property_name*, SMALL SLIDING
slave_surface_name, *master_surface_name*

You can also specify the contact discretization directly:

*CONTACT PAIR, INTERACTION=*interaction_property_name*, SMALL SLIDING, TYPE=NODE TO SURFACE
slave_surface_name, *master_surface_name*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Surface-to-surface contact (Standard)**: select the master surface, click **Surface** or **Node Region**, select the slave 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 slave surface precludes the use of surface-to-surface discretization.

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

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

Using symmetric master-slave contact pairs to improve contact modeling

For node-to-surface contact it is possible for master surface nodes to penetrate the slave surface without resistance with the strict master-slave algorithm used by Abaqus/Standard. This penetration tends to occur if the master surface is more refined than the slave surface or a large contact pressure develops between soft bodies. Refining the slave surface mesh often minimizes the penetration of the master surface nodes. If the refinement technique does not work or is not practical, a symmetric master-slave 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 master and slave surface for the two contact pairs. This method causes Abaqus/Standard to treat each surface as a master 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 master-slave 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 master surface, click **Surface**, select the slave surface
Copy this interaction to a new interaction, and edit the new interaction. In the interaction editor, click **Switch** to reverse the master and slave surfaces.

Limitations of symmetric master-slave contact pairs

Using symmetric master-slave contact pairs can lead to overconstraint problems when very stiff or "hard" contact conditions are enforced. See "Contact constraint enforcement methods in Abaqus/Standard," Section 36.1.2, for a discussion of overconstraints and alternate constraint enforcement methods.

For softened contact conditions, use of symmetric master-slave 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 master-slave contact pairs effectively double the overall contact stiffness if a linear pressure-overclosure relationship is specified.

Likewise, use of symmetric master-slave contact pairs will cause deviations from the friction model if an optional shear stress limit is specified (see "Using the optional shear stress limit" in "Frictional behavior," Section 35.1.5), 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 master-slave contact pairs. In this case both surfaces at the interface act as slave 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 master and slave surfaces per contact pair and not just the slave surface where constraints are formed. Consequently, two result sets are available per surface of a symmetric master-slave contact pair; once when a surface acts as a slave and once as a master. 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 master or as a slave 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 master-slave 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 master and a slave, 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.

Since a node of a self-contacting surface can be both a slave node and a member of the master surface, contact behavior is very similar to symmetric master-slave contact pairs, and the issues discussed in "Using symmetric master-slave contact pairs to improve contact modeling" above apply to three-dimensional self-contact. In the special case of two-dimensional self-contact the nodes adjacent to a vertex where a surface folds over on itself follow a strict master-slave algorithm to avoid overconstraints. Abaqus/Standard automatically applies some numerical "softening" to the contact conditions with most self-contact formulations. See "Contact constraint enforcement methods in Abaqus/Standard," Section 36.1.2, for a discussion of the numerical constraint enforcement methods used with self-contact.

Selecting surfaces used in contact pairs

Methods for creating surfaces are discussed in "Element-based surface definition," Section 2.3.2; "Node-based surface definition," Section 2.3.3; and "Analytical rigid surface definition," Section 2.3.4; 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," Section 36.1.1. Additional considerations for surfaces used in contact definitions are discussed below.

Orientation considerations for shell-like surfaces

Abaqus/Standard requires master 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" in "Contact formulations in Abaqus/Standard," Section 36.1.1, for details). This requires that you consider the proper orientation for master surfaces defined on elements, such as shells and membranes, that have positive and negative directions. For node-to-surface contact the orientation of slave surface normals is irrelevant, but for surface-to-surface contact the orientation of single-sided slave 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 master and slave 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," Section 34.3.4), the slave surface should be on the same side of the master surface as the outward normal. If, in the initial configuration, the slave surface is on the opposite side of the master 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 34.3.1–1 illustrates the proper and improper specification of a master surface's outward normal.
- Contact will be ignored with surface-to-surface discretization if single-sided slave and master surfaces have normal directions that are in approximately the same direction (for example, contact will not be enforced if the dot product of the slave and master surface normals is positive).

The following output from a data check analysis (see "Abaqus/Standard, Abaqus/Explicit, and Abaqus/CFD execution," Section 3.2.2) can be useful in identifying incorrectly oriented master 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.

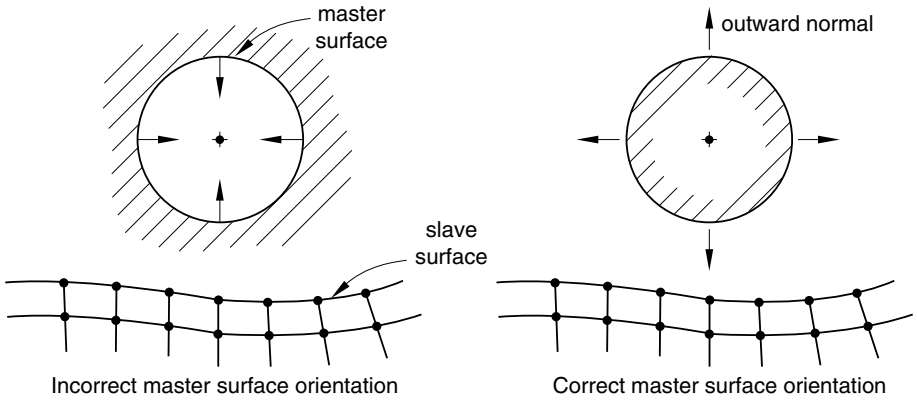


Figure 34.3.1–1 Example of proper and improper master surface orientation.

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 34.3.1–1. As indicated in this table, the connectivity restrictions are sometimes different for master and slave surfaces. Self-contact surfaces act as both master and slave surfaces; therefore, if a restriction applies to either a master or slave surface, it also applies to self-contact. The potential connectivity restrictions referred to in Table 34.3.1–1 are described below:

Table 34.3.1–1 Summary of which connectivity characteristics of element-based surfaces are allowed for various contact formulations.

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

- Discontinuous surfaces: Discontinuous contact surfaces are allowed in many cases, but the master 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 34.3.1–2 shows examples of continuous surfaces, whereas Figure 34.3.1–3 and Figure 34.3.1–4 show examples of discontinuous surfaces. Figure 34.3.1–5 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 master surface for finite-sliding, node-to-surface contact.

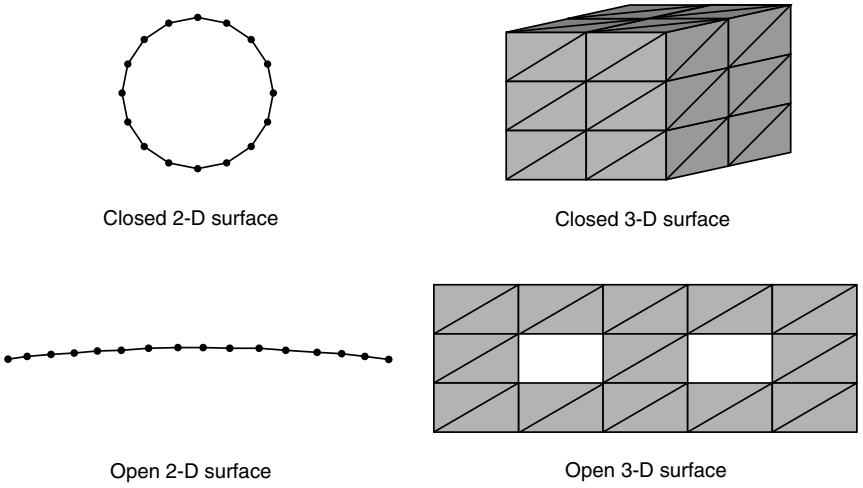


Figure 34.3.1–2 Examples of continuous surfaces.



Figure 34.3.1–3 Example of a discontinuous 2-D surface.

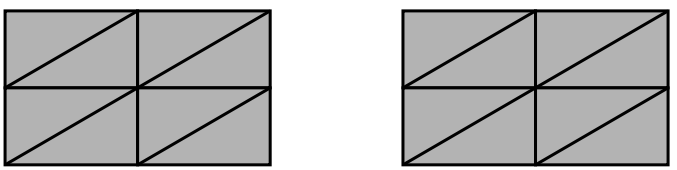


Figure 34.3.1–4 Example of a discontinuous 3-D surface.

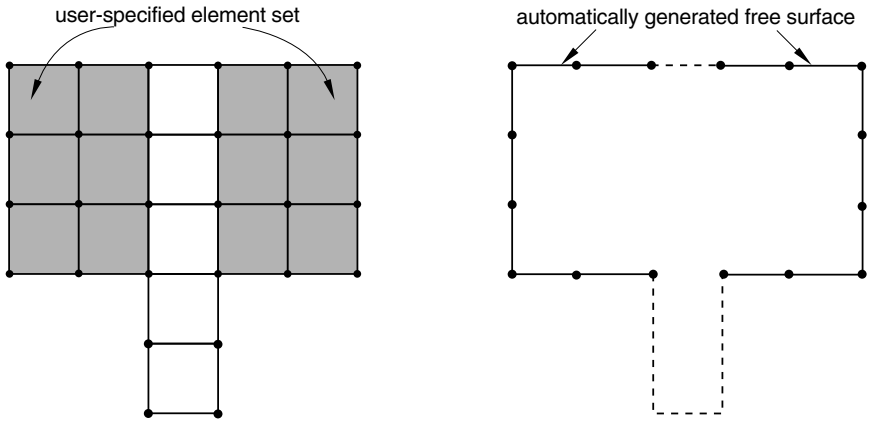


Figure 34.3.1–5 Example of a discontinuous surface resulting from automatic free surface generation with a disjoint element set.

- Portions of three-dimensional surfaces joined at only one node: The finite-sliding, node-to-surface contact formulation also does not allow three-dimensional master surface faces to be joined at a single node (they must be joined across a common element edge). Figure 34.3.1–6 shows an example of a surface with two faces connected by a single node.

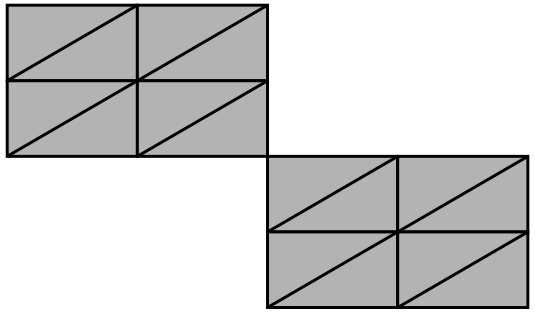


Figure 34.3.1–6 Example of a 3-D surface with two faces sharing a single node.

- Surfaces with T-intersections: In some cases a contact surface cannot have more than two surface faces sharing a common master node in two dimensions or a common master edge in three dimensions. For example, Figure 34.3.1–7 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 slave node in two dimensions or a common edge in three dimensions for node-to-surface formulations, the slave faces must be single-sided, which precludes the most common T-intersection cases for node-to-surface formulations.

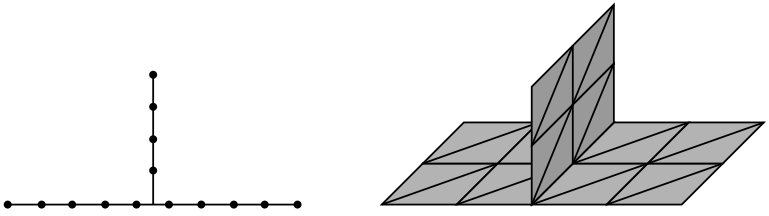


Figure 34.3.1–7 Examples of surfaces with T-intersections.

Analytical rigid surfaces

Analytical rigid surfaces are often effective for efficiently modeling curved, rigid geometries, as discussed in "Analytical rigid surface definition," Section 2.3.4. 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," Section 2.3.2).

Three-dimensional beam and truss surfaces

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

Edge-based surfaces

Edge-based surfaces ("Element-based surface definition," Section 2.3.2) 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," Section 35.1.2; "Thermal contact properties," Section 35.2.1; or "Electrical contact properties," Section 35.3.1.

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.

You cannot remove tied contact pairs from a simulation (see "Defining tied contact in Abaqus/Standard," Section 34.3.7).

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 may 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
slave_surface, master_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," Section 34.3.4). 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
slave_surface, master_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," Section 34.3.4.

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," Section 37.1.1.

For more detailed discussions of variables associated with thermal, electrical, and pore fluid analyses, see the sections on the related contact properties in Chapter 35, "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 slave or master nodes. In such cases the nodal values are generally affected by more than one contact constraint. Other nodal contact variables are available only at nodes acting as slave nodes. In these cases the value at each slave node reflects a value associated with a particular

contact constraint. Most contact output to the data (`.dat`) and results (`.fil`) files is associated with individual constraints.

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," Section 12.3.2, 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.
- To minimize contact stress noise within a region of active contact, Abaqus/Standard computes nodal contact stresses as weighted averages of values associated with active contact constraints in which a node participates. Some filtering is applied to reduce the contact stress values reported for nodes on the fringe of the active contact region (that only weakly participate in contact constraints), but this filtering is not "perfect," which can result in the contact zone size appearing somewhat exaggerated. Similarly, contact status output will also be 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 slave surface. 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," Section 4.2.1.

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

- request output associated with a given contact pair;
- request output associated with a given slave surface, including contributions from all of the contact pairs to which the slave surface belongs; and
- limit the output by specifying a node set containing a subset of the nodes on the slave 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" in "Output to the data and results files," Section 4.1.2.
- To request output to the output database (`.odb`) file, see "Surface output in Abaqus/Standard and Abaqus/Explicit" in "Output to the output database," Section 4.1.3.

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 slip directions defined on the surfaces. The definition of slip directions is explained in "Local tangent directions on a surface" in "Contact formulations in Abaqus/Standard," Section 36.1.1. 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 slip 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 slip direction. For example, CSHEAR1 is the frictional shear stress component in the first slip direction, while CSHEAR2 is the frictional shear stress component in the second slip 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 slip direction. The incremental relative nodal displacement vector measures the motion of a slave node relative to the motion of the master surface. The incremental slip is accumulated only when the slave node is contacting the master 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," Section 5.1.1 of the Abaqus Theory Manual; "Finite-sliding interaction between deformable bodies," Section 5.1.2 of the Abaqus Theory Manual; and "Finite-sliding interaction between a deformable and a rigid body," Section 5.1.3 of the Abaqus Theory Manual.

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

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," Section 1.1.20 of the Abaqus Example Problems Manual). The maximum torque, T, is defined as

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

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.

34.3.2 ASSIGNING SURFACE PROPERTIES FOR CONTACT PAIRS IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- *CONTACT PAIR
- "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining self-contact," Section 15.13.7 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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 34.3.2–1.

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," Section 35.1.2) is used to avoid overconstraining slave nodes. The following input is used:

```
*SURFACE, NAME=TOP_RIG_SURF
TOP_RIG_ELS,
*SURFACE, NAME=SHELL_TOP_SURF
```

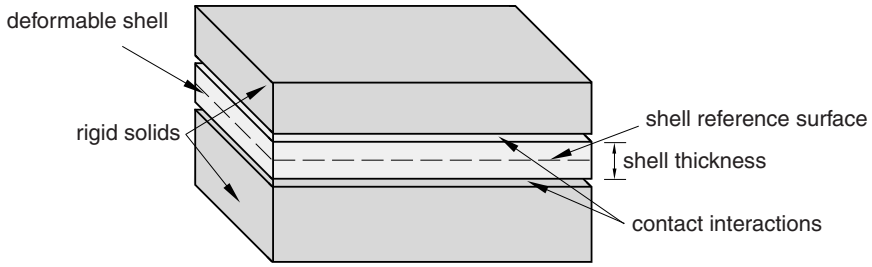


Figure 34.3.2–1 Shell pinched between two rigid bodies.

```
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," Section 36.1.1, and "Smoothing contact surfaces in Abaqus/Standard," Section 36.1.3.

34.3.3 ASSIGNING CONTACT PROPERTIES FOR CONTACT PAIRS IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Contact interaction analysis: overview," Section 34.1.1
- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- *CONTACT PAIR
- *SURFACE INTERACTION
- "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining self-contact," Section 15.13.7 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Using contact and constraint detection," Section 15.16 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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 "Mechanical contact properties: overview," Section 35.1.1, 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," Section 36.1.2.

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

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 34.3.3–1 shows the mesh used in this example. For purposes of this example, the surface **ASURF** is the slave 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.

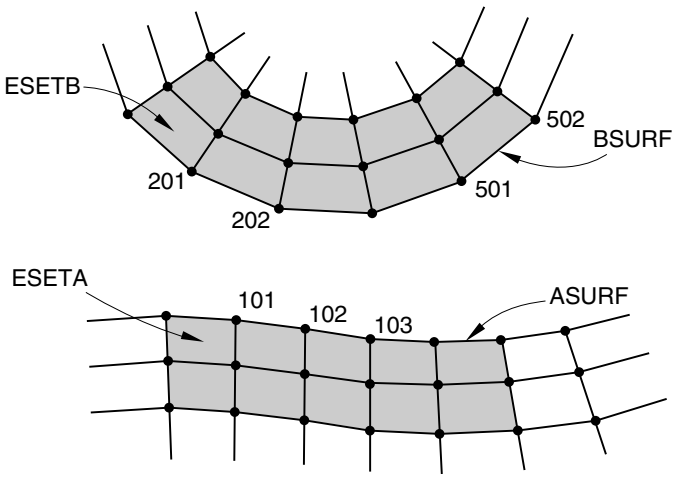


Figure 34.3.3–1 Mechanical surface interaction with friction and finite sliding.

```
*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
```

34.3.4 MODELING CONTACT INTERFERENCE FITS IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- *CONTACT INTERFERENCE
- "Specifying interference fit options" in "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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 "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5.

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 34.3.4–1 illustrates a typical interference fit problem. 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

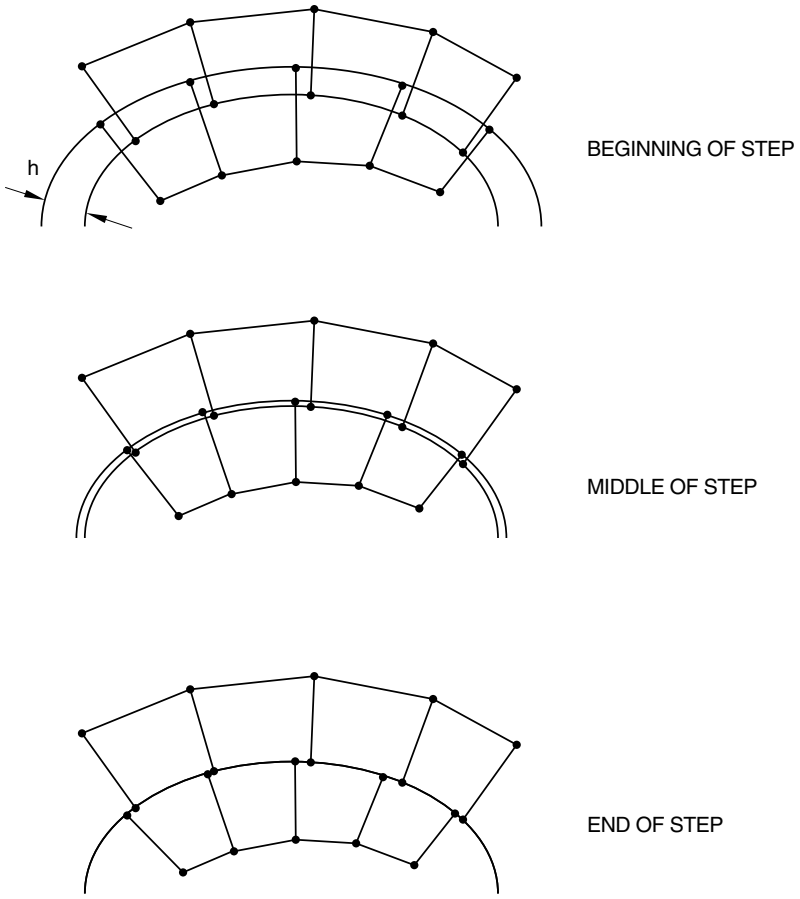


Figure 34.3.4–1 Interference fit with contact surfaces.

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
slave surface, *master 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 slave 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," Section 32.1.2, 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," Section 6.2.4) is used.

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

Abaqus/CAE Usage: Interaction module: interaction editor: **Interference Fit**: **Gradually remove slave 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 slave surface or every slave node in the specified set of contact elements. The concepts of slave 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 slave nodes reported in the message file when you request detailed contact printout. Thus, any slave node that penetrates the master 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 slave 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 slave 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 slave 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 34.3.4–2 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. 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 slave 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.

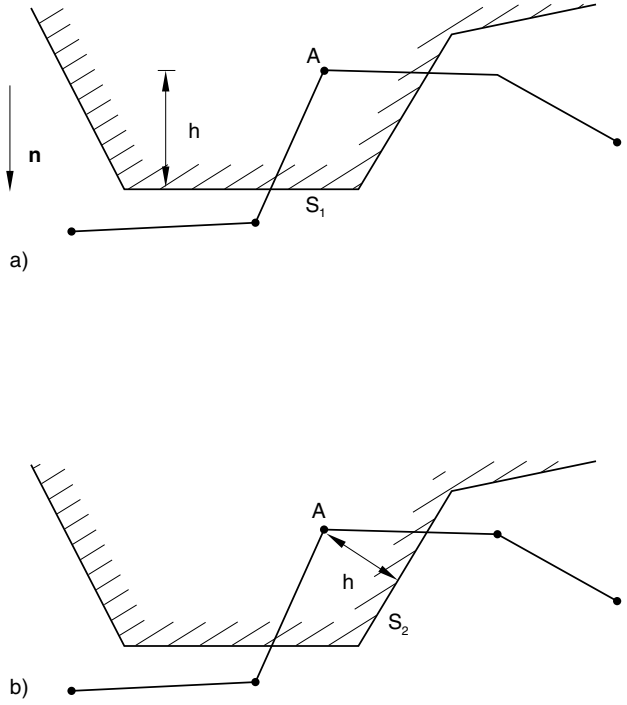


Figure 34.3.4–2 Effect of direction definition on interference accommodation: a) with direction, b) without direction.

In case (b) slave 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
slave surface, *master 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 slave 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 slave 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 slave facet normal

directions; using node-to-surface contact, overclosures tend to be resolved along the master surface normal directions. Figure 34.3.4–3 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.

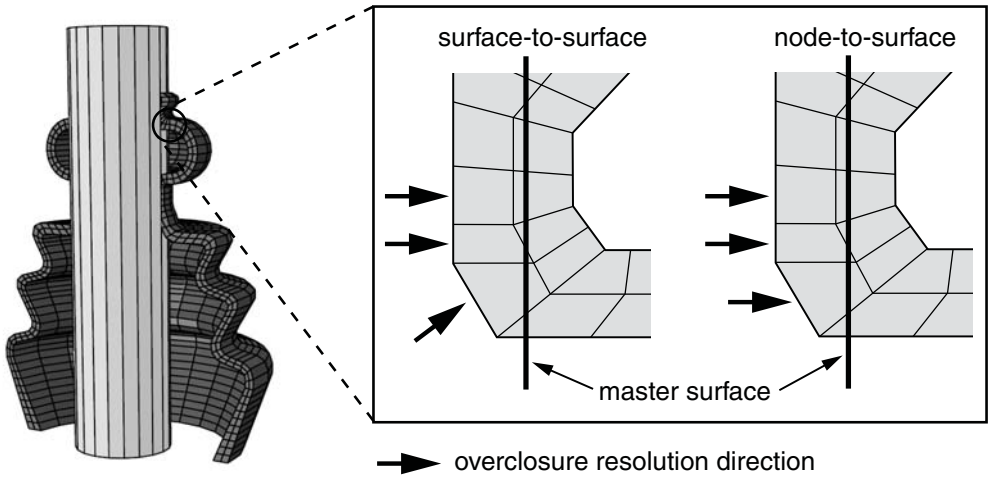


Figure 34.3.4–3 Comparison of contact formulations in an example with a large interference fit.

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" in "Frictional behavior," Section 35.1.5).

34.3.5 ADJUSTING INITIAL SURFACE POSITIONS AND SPECIFYING INITIAL CLEARANCES IN Abaqus/Standard CONTACT PAIRS

Products: Abaqus/Standard Abaqus/CAE

References

- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- "Modeling contact interference fits in Abaqus/Standard," Section 34.3.4
- "Defining tied contact in Abaqus/Standard," Section 34.3.7
- "Contact formulations in Abaqus/Standard," Section 36.1.1
- *CLEARANCE
- *CONTACT PAIR
- "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Using contact and constraint detection," Section 15.16 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

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 slave surface so that they precisely contact the master 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; and
- cannot be used with self-contact or symmetric master-slave contact.
- 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," Section 36.1.1).

In addition to adjusting two surfaces into precise contact, Abaqus/Standard offers various methods to define the initial clearances between two surfaces precisely in both magnitude and direction. Responses to negative clearances, or interference fits, are discussed in "Modeling contact interference fits in Abaqus/Standard," Section 34.3.4.

Adjusting the surfaces in a contact pair

You can have Abaqus/Standard adjust the position of the slave surface of a contact pair by specifying either a floating point value a for the depth of an "adjustment zone" around the master surface or a node set label.

Abaqus/Standard does not adjust the nodes on the slave surface by default for contact pairs; rather initial overclosures are treated as interference fits by default for contact pairs.

Comments unique to surface-to-surface contact

The following points apply to contact pairs with surface-to-surface discretization (see "Contact formulations in Abaqus/Standard," Section 36.1.1, for further discussion of surface-to-surface discretization):

- Strain-free adjustments to slave node positions may not result in exactly zero gap with respect to the master surface as measured at a slave node. The adjustments are made to achieve zero gap between the surfaces in an average sense in a region near each slave node within the adjustment zone.
- The magnitude of strain-free adjustments is limited to half the typical facet length. For instances of initial overclosures exceeding this limit, an allowable penetration equal to the initial overclosure is stored for the associated contact constraints such that penetrations deeper than the initial overclosure are resisted during the analysis, but penetrations less than the initial overclosure are not resisted.
- Strain-free adjustments will occur for some slave nodes outside the adjustment zone if a significant portion of a slave face (or segment in two dimensions) to which it is attached is within the adjustment zone.

The discussion in the remainder of this section applies directly to node-to-surface contact discretizations (for which contact is enforced at discrete points—slave nodes) but should be considered within the context of the above points for surface-to-surface contact discretizations.

Using an "adjustment zone" when adjusting surfaces

When you specify a, the depth of the "adjustment zone," Abaqus/Standard forms an adjustment zone extending a distance a from the master surface. Abaqus/Standard measures the distance along the master surface normals that pass through the nodes of the slave surface. Any nodes on the slave surface that are within the "adjustment zone" in the initial geometry of the model are moved precisely onto the master surface. The motion of these slave 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 34.3.5–1 and Figure 34.3.5–2. If you specify a negative value for a, Abaqus/Standard will issue an error message.

Input File Usage: ∗CONTACT PAIR, ADJUST=a
slave_surface, master_surface
...

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Specify tolerance for adjustment zone**: a

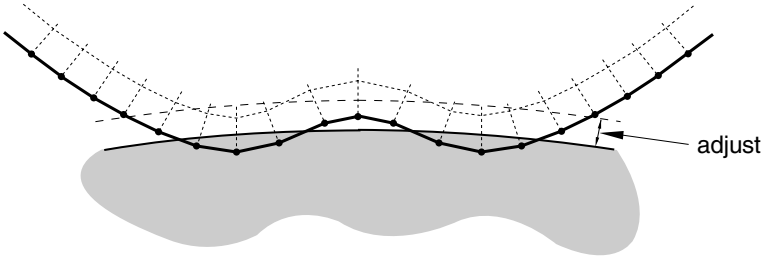


Figure 34.3.5–1 Initial configuration of the contact surfaces showing the "adjustment zone." The slave surface is in bold.

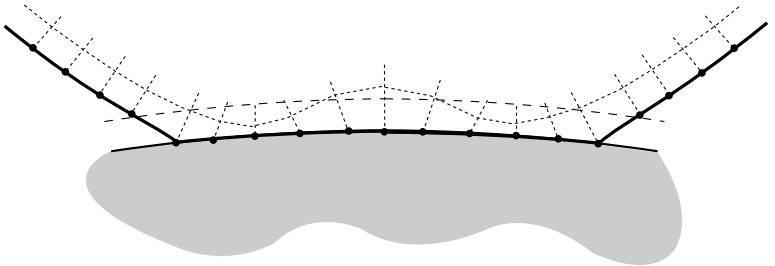


Figure 34.3.5–2 Configuration of the contact surfaces after the adjustment. Nodes within the adjustment zone and overclosed nodes have been moved.

Adjusting overclosed slave nodes using an adjustment zone

When you specify the depth of the adjustment zone, Abaqus/Standard moves any slave nodes penetrating the master surface in the initial configuration so that they just contact the master surface. Specifying a value of 0.0 for a causes Abaqus/Standard to adjust only those slave nodes that are penetrating the master surface. Figure 34.3.5–3 shows the effect of specifying a=0.0 in the example shown in Figure 34.3.5–1. If you do not have Abaqus/Standard adjust the position of the slave surface, slave nodes that are overclosed in the initial configuration will remain overclosed at the start of the simulation, which may cause convergence problems.

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 slave nodes should be adjusted and specifying a may cause the inappropriate adjustment of other slave nodes. Abaqus/Standard adjusts only those nodes on the slave surface belonging to the node set. The node set can contain nodes that are not on the slave surface at all: Abaqus/Standard will ignore them and adjust only the nodes in the node set that are part of the slave surface.

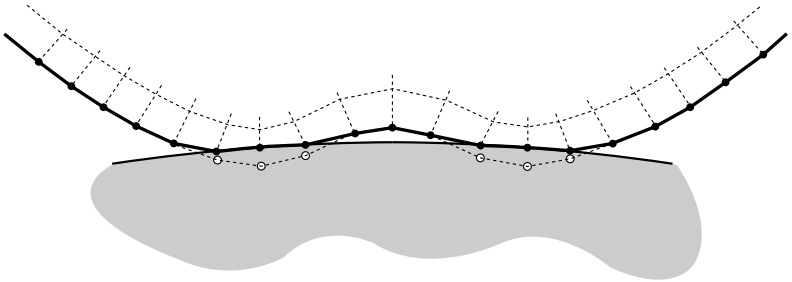


Figure 34.3.5–3 Adjusted configuration of contact surfaces when a=0.

Abaqus/Standard moves any slave nodes in the specified node set regardless of how far they are from the master surface. The adjustments of the nodes from their initial configurations do not create strains in the elements forming the slave surface. If Abaqus/Standard adjusts slave nodes that are far from the master surface, the elements may become poorly shaped, which can cause convergence difficulties.

Input File Usage: ∗CONTACT PAIR, ADJUST=*node_set_label*
slave_surface, master_surface
...

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Adjust slave nodes in set:** *node_set_label*

Adjusting overclosed slave nodes using a node set label

Because Abaqus/Standard adjusts only the slave nodes in the specified node set, any overclosed slave 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 slave 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 slave surface.

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 slave nodes and later as master 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.

When to adjust contact surface pairs

There are several instances when adjusting the surfaces in a contact pair is required or strongly recommended:

- When tying two surfaces together for the duration of the analysis (see "Defining tied contact in Abaqus/Standard," Section 34.3.7).
- When using small- or infinitesimal-sliding contact (see "Contact formulations in Abaqus/Standard," Section 36.1.1).
- When specifying a precise initial clearance or initial overclosure for the contact surfaces by defining an allowable contact interference (see "Alternative methods for specifying precise initial clearances or overclosures" below).

Defining a precise 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 slave 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.

The initial clearance or overclosure value calculated at every slave node (based on the coordinates of the slave node and the master surface) is overwritten by the value that you specify. This procedure is performed internally, and it does not affect the coordinates of the slave nodes. If you define a clearance, Abaqus/Standard will treat the two surfaces as not being in contact, regardless of their nodal coordinates. If you define an overclosure, Abaqus/Standard will treat the two surfaces as an interference fit and attempt to resolve the overclosure in the first increment. If the defined overclosure is large, you may need to specify an allowable interference that is ramped off over several increments. See "Modeling contact interference fits in Abaqus/Standard," Section 34.3.4, for further discussion of interference fits.

You can define initial clearance or overclosure values only for small-sliding contact ("Contact formulations in Abaqus/Standard," Section 36.1.1). 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.

Specifying a uniform clearance or overclosure for the surfaces

You can specify a uniform clearance or overclosure for a contact pair by identifying the master and slave 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.

Input File Usage: *CLEARANCE, SLAVE=*surface_name*, MASTER=*surface_name*, VALUE=h_0

Abaqus/CAE Usage: Interaction module: contact interaction editor: **Clearance**: **Initial clearance: Uniform value across slave surface:** h_0

Specifying spatially varying clearances or overclosures for the surfaces

Alternatively, you can specify spatially varying clearances or overclosures for a contact pair by identifying the master and slave surfaces of the contact pair and providing a table of data specifying the clearance at a single node or a set of nodes belonging to the slave surface. Any slave surface node that is not identified will use the clearance that Abaqus/Standard calculates from the initial geometry of the surfaces.

Input File Usage: ∗CLEARANCE, SLAVE=*surface_name*, MASTER=*surface_name*, TABULAR
node number or node set label, clearance value

Repeat the data line as often as necessary.

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 from an external file.

Input File Usage: ∗CLEARANCE, SLAVE=*surface_name*, MASTER=*surface_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/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,” Section 36.1.1. When specifying spatially varying clearances or overclosures, you can redefine the contact direction that Abaqus/Standard uses with each slave node by specifying the components of this vector. The vector must be defined in the global Cartesian coordinate system, and it should define the master surface’s desired outward normal direction.

Input File Usage: ∗CLEARANCE, SLAVE=*surface_name*, MASTER=*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 threaded bolt connections (see “Generating the contact normal directions for a threaded bolt connection automatically” below).

Generating the contact normal directions for a threaded bolt connection automatically

Alternatively, for a single-threaded bolt connection the contact normal directions for each slave 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. Either the bolt or bolt hole can be a master or slave surface. However, the vector defining the axis of the bolt or bolt hole must be chosen appropriately.

For example, when the bolt surface is chosen to be the master surface, the vector should be oriented to point from the tip of the bolt to the head of the bolt if the bolt is in tension and from the head to the tip if the bolt is in compression. If the bolt surface is chosen to be the slave surface and the bolt is in tension, the bolt axis should be flipped (i.e., from the head to the tip) and a negative half-thread angle should be specified. An incorrect bolt axis direction will not engage the contact interaction, and the surfaces will be unconstrained. You should check the stresses in the bolt to make sure that the contact is engaged.

Input File Usage: ∗CLEARANCE, SLAVE=*surface_name*, MASTER=*surface_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 slave 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*

Visualizing the precise initial clearances or overclosures

Abaqus/Standard does not adjust the coordinates of the slave 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 may 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.

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 the surfaces in a contact pair”) 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, v, for the contact pair (see “Modeling contact interference fits in Abaqus/Standard,” Section 34.3.4). 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*
slave_surface, master_surface
∗AMPLITUDE, NAME=*amplitude_name*
∗CONTACT INTERFERENCE, AMPLITUDE=*amplitude_name*
slave_surface, master_surface, 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**: 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, 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 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, v. When viewed in Abaqus/CAE, the surfaces start the simulation with coordinates that have them exactly touching, but the specified interference 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 $|v|$, the surfaces will behave as if they are precisely in contact.

34.3.6 ADJUSTING CONTACT CONTROLS IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- *CONTACT CONTROLS
- *CONTACT PAIR
- "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining self-contact," Section 15.13.7 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Specifying contact controls in an Abaqus/Standard analysis," Section 15.13.8 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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 the 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, SLAVE=*slave surface*, MASTER=*master 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, SLAVE=*slave surface*,
MASTER=*master 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 slave nodes, in the same manner as contact damping (see "Contact damping," Section 35.1.3). 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 post-analysis 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. This is particularly useful if Abaqus is not able to calculate a sensible damping value. For example, this may be the case if the slave surface is a node-based surface, in which case the properties of the underlying elements are not available. 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*

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 slave 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 master 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," Section 36.1.2.

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*, SLAVE=*slave surface*, MASTER=*master surface*

Abaqus/CAE Usage: Modifying the tangential penalty stiffness 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," Section 36.1.1, for a discussion of these contact formulations).

Some second-order element types are not well-suited for underlying the slave 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 34.3.6–1, 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. This ambiguous nature of the nodal forces in second-order elements can cause Abaqus/Standard to alter its internal contact logic inadequately. Slave 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 slave 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 option to 10-node second-order tetrahedral elements (C3D10, C3D10I, 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" in "Solid (continuum) elements," Section 27.1.1. 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 slave facets associated with 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,

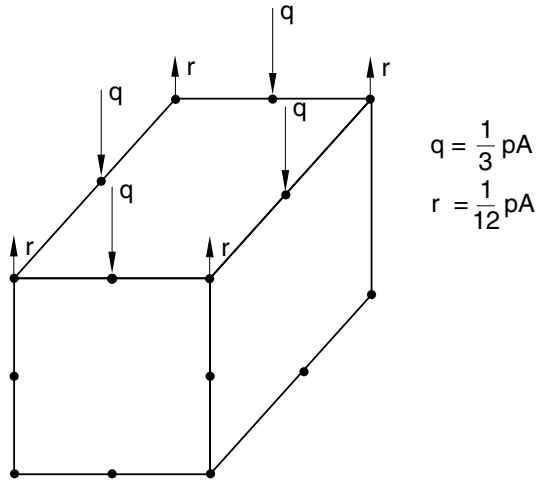


Figure 34.3.6–1 Equivalent nodal loads produced by a constant pressure on the second-order element face in "hard" contact simulations.

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.

By default, Abaqus/Standard does not automatically add midface nodes to second-order serendipity elements that form a slave 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 slave 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 slave faces of non-modified elements and 8-node slave 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 slave surface.

Input File Usage: ∗CONTACT PAIR, INTERACTION=*interaction_property_name*, SUPPLEMENTARY CONSTRAINTS=SELECTIVE
slave_surface_name, *master_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
slave_surface_name, *master_surface_name*

Use the following option to forgo supplementary contact constraints:

∗CONTACT PAIR, INTERACTION=*interaction_property_name*, SUPPLEMENTARY CONSTRAINTS=NO
slave_surface_name, *master_surface_name*

Abaqus/CAE Usage: For a node-to-surface contact formulation:

Interaction module: **Create Interaction**: **Surface-to-surface contact (Standard)**: select the master surface; click **Surface**; select the slave 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 slave 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 slave surface for surface-to-surface contact pairs:

∗CONTACT PAIR, TYPE=SURFACE TO SURFACE, SLIDING
TRANSITION=ELEMENT ORDER SMOOTHING
slave_surface_name, *master_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
slave_surface_name, *master_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
slave_surface_name, *master_surface_name*

Abaqus/CAE Usage: You cannot change the default contact force redistribution in Abaqus/CAE.

34.3.7 DEFINING TIED CONTACT IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5
- *CONTACT PAIR
- "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Using contact and constraint detection," Section 15.16 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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 slave surface to have the same value of displacement, temperature, pore pressure, or electrical potential as the point on the master surface that it contacts;
- allows for rapid transitions in mesh density within the model;
- requires the adjustment of the contact pair surfaces; and
- cannot be used with self-contact or symmetric master-slave contact.

It is preferable to use the surface-based tie constraint capability instead of tied contact (see "Mesh tie constraints," Section 33.3.1, for details).

Defining tied contact for a contact pair

To "tie" the surfaces of a contact pair together for an analysis, you must also adjust the surfaces because, as described below, it is very important that the tied surfaces be precisely in contact at the start of the simulation. See "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5, for details on adjusting surfaces. As always, you must associate the contact pair with a contact interaction property definition.

Input File Usage: *CONTACT PAIR, TIED, ADJUST=α or *node_set_label*, INTERACTION=*name*

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

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 slave nodes are within the adjustment zone (see "Adjusting the surfaces in a contact pair" in "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5), accounting for any shell or membrane thickness by default. Abaqus/Standard then adjusts these slave nodes' positions into a zero-penetration state and forms constraints between these slave nodes and the surrounding nodes on the master surface. The constraints are formed with 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 "Defining contact pairs in Abaqus/Standard," Section 34.3.1, and "Assigning surface properties for contact pairs in Abaqus/Standard," Section 34.3.2).

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," Section 36.1.2). 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," Section 35.1.2) 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 slave nodes to the master surface unless they are precisely in contact with the master surface at the start of the analysis. Any slave 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 master surface. In mechanical simulations an unconstrained slave node can penetrate the master surface freely. In a thermal, electrical, or pore pressure simulation an unconstrained slave node will not exchange heat, electrical current, or pore fluid with the master surface.

To avoid such unconstrained nodes in tied contact pairs, use the capability for adjusting the surfaces of a contact pair described in "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5. This capability moves slave nodes onto the master surface before Abaqus/Standard checks for the initial contact state. It is intended only for nodes that are close to the master surface and is not intended to correct large errors in the mesh geometry.

Checking that slave nodes are constrained

Abaqus/Standard prints a table in the data (`.dat`) file identifying the predominant slave node and other nodes involved in each constraint. If Abaqus/Standard cannot form a constraint for a given slave node acting as a predominant slave node, either because it is not in contact with the master surface or it cannot "see" the master surface, it will issue a warning message in the data file. For an explanation of when a slave node would not "see" a master surface and how to correct this problem, see "Contact formulations in Abaqus/Standard," Section 36.1.1. 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.

34.3.8 EXTENDING MASTER SURFACES AND SLIDE LINES

Product: Abaqus/Standard

References

- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- "Common difficulties associated with contact modeling in Abaqus/Standard," Section 37.1.2
- *CONTACT PAIR
- *SLIDE LINE

Overview

Extending the master surface or a slide line:

- can prevent nodes from "falling off" or getting trapped behind the master surface (or slide line) in finite-sliding problems;
- allows the slave node to find a master surface when the slave node has no intersection with the master 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; and
- applies only to contact pairs that use a node-to-surface discretization.

Extending the master surface for small-sliding, node-to-surface contact

If a slave node cannot find an intersection with the master surface at the start of the analysis, it will be free to penetrate the master surface because no local tangent plane will be formed. This type of problem, which typically occurs for node-to-surface contact when the slave node is aligned with the edge of the master surface, is illustrated in Figure 34.3.8–1 and may be caused by numerical roundoff errors when a preprocessor is used to generate the nodal coordinates. Cases such as that shown in Figure 34.3.8–1 are not problematic for the small-sliding, surface-to-surface formulation because the constraint formulation considers the region of the slave surface near a slave node.

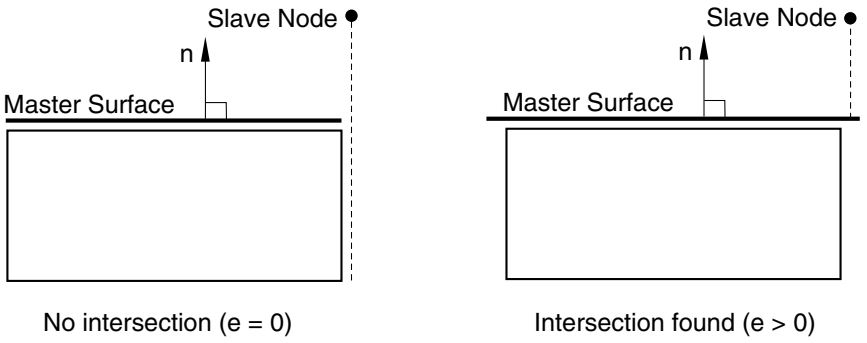


Figure 34.3.8–1 Slave node fails to find an intersection with the master surface for small-sliding, node-to-surface contact if $e=0$.

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 34.3.8–2). 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.

Input File Usage: ∗CONTACT PAIR, SMALL SLIDING, EXTENSION ZONE=e

Extending the master surface or slide line in finite-sliding, node-to-surface contact

To prevent slave nodes from "falling off" or getting trapped behind the master surface, an open surface or slide line can be extended 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 34.3.8–2). 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 slave faces, and "falling off" will not occur until nearly the entire slave facet slides off the master 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 slave 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

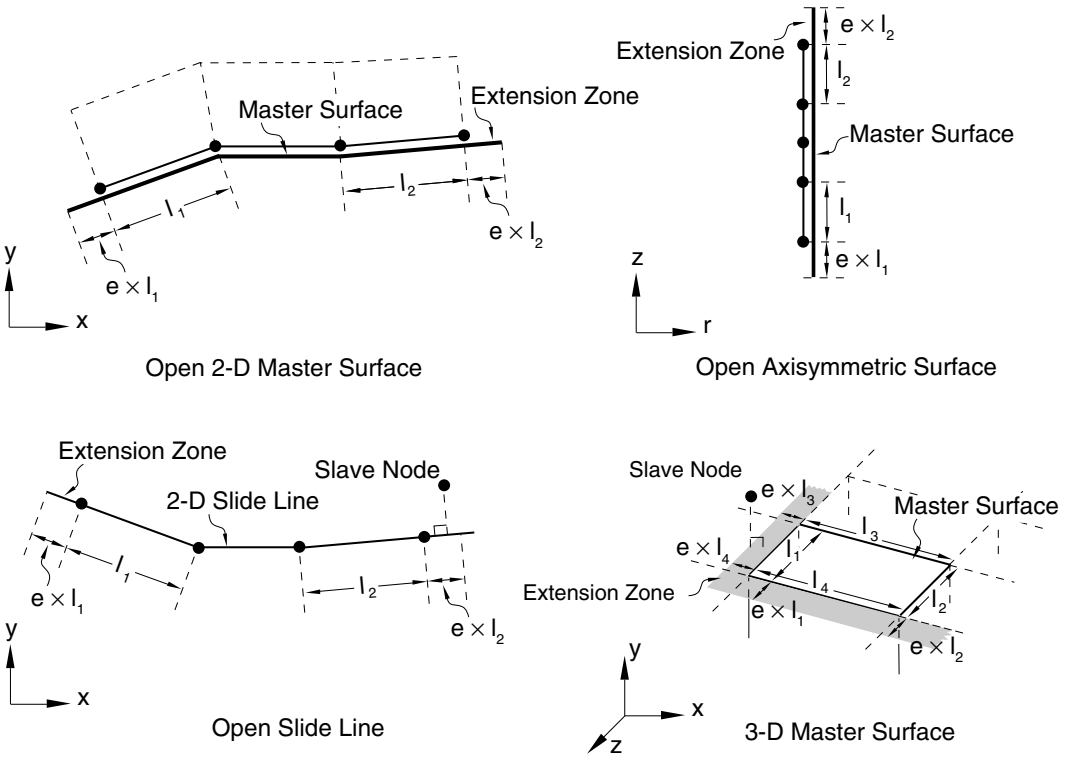


Figure 34.3.8–2 Definition of size of extension zone.

34.3.9 CONTACT MODELING IF SUBSTRUCTURES ARE PRESENT

Product: Abaqus/Standard

References

- "Element-based surface definition," Section 2.3.2
- "Node-based surface definition," Section 2.3.3
- "Using substructures," Section 10.1.1
- "Membrane elements," Section 28.1.1
- "Surface elements," Section 31.7.1
- "Contact interaction analysis: overview," Section 34.1.1
- "Defining contact pairs in Abaqus/Standard," Section 34.3.1

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 master or slave 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 master 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," Section 2.3.2), 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," Section 31.7.1, for more information about surface elements.

Figure 34.3.9–1 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.

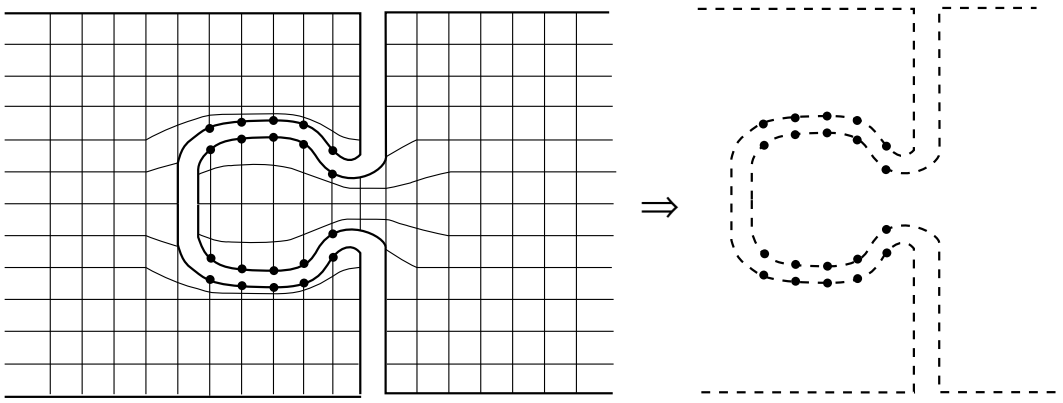


Figure 34.3.9–1 Substructuring in a contact simulation.

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 34.3.9–1 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 slave 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 slave surface (see "Three-dimensional surfaces with second-order faces and a node-to-surface formulation" in "Common difficulties associated with contact modeling in Abaqus/Standard," Section 37.1.2, 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 slave 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 slave surface of a contact pair, the retained nodes can be included in a node-based surface (see "Node-based surface definition," Section 2.3.3). 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," Section 38.2.1) 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.

34.3.10 CONTACT MODELING IF ASYMMETRIC-AXISYMMETRIC ELEMENTS ARE PRESENT

Product: Abaqus/Standard

References

- "Slide line contact elements," Section 38.4.1
- "Rigid surface contact elements," Section 38.5.1
- *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," Section 27.1.7, and "Axisymmetric shell elements with nonlinear, asymmetric deformation," Section 28.6.10) 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.

34.4 Defining general contact in Abaqus/Explicit

- "Defining general contact interactions in Abaqus/Explicit," Section 34.4.1
- "Assigning surface properties for general contact in Abaqus/Explicit," Section 34.4.2
- "Assigning contact properties for general contact in Abaqus/Explicit," Section 34.4.3
- "Controlling initial contact status for general contact in Abaqus/Explicit," Section 34.4.4
- "Contact controls for general contact in Abaqus/Explicit," Section 34.4.5

34.4.1 DEFINING GENERAL CONTACT INTERACTIONS IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Contact interaction analysis: overview," Section 34.1.1
- *CONTACT
- *CONTACT INCLUSIONS
- *CONTACT EXCLUSIONS
- "Defining general contact," Section 15.13.1 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Abaqus/Explicit provides two algorithms for modeling contact and interaction problems: the general contact algorithm and the contact pair algorithm. See "Contact interaction analysis: overview," Section 34.1.1, 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 may 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 (i.e., some interactions can be modeled with the general contact algorithm, while others are modeled with the contact pair algorithm);
- can be used only with 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 (i.e., 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," Section 34.4.2);

- the mechanical contact property models (“Assigning contact properties for general contact in Abaqus/Explicit,” Section 34.4.3);
- the contact formulation (“Contact formulation for general contact in Abaqus/Explicit,” Section 36.2.1);
- the initial clearance between contact surfaces (“Controlling initial contact status for general contact in Abaqus/Explicit,” Section 34.4.4); and
- the algorithmic contact controls (“Contact controls for general contact in Abaqus/Explicit,” Section 34.4.5).

Surfaces used for general contact

The general contact algorithm allows for very general characteristics in the surfaces that it uses, as discussed in “Contact interaction analysis: overview,” Section 34.1.1. For detailed information on defining surfaces in Abaqus/Explicit for use with the general contact algorithm, see “Element-based surface definition,” Section 2.3.2; “Node-based surface definition,” Section 2.3.3; “Analytical rigid surface definition,” Section 2.3.4; “Eulerian surface definition,” Section 2.3.5; and “Operating on surfaces,” Section 2.3.6. Two-dimensional surfaces cannot be used with the general contact algorithm.

A convenient method of specifying the contact domain is using cropped surfaces. Such surfaces can be used to perform “contact in a box” by using a contact domain that is enclosed in a specified rectangular box in the original configuration. For more information, see “Operating on surfaces,” Section 2.3.6.

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. 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.

Considerations for edge-to-edge contact

The 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. For example, contact between beam segments and shell perimeter edges (see Figure 34.4.1–1) usually is detected only as edge-to-edge contact. 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. The contact edges representing beam and truss elements have a circular cross-section, regardless of the actual cross-section of the beam or truss element. The radius of a contact edge representing a 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 beams with circular cross-sections, the radius of the contact edge is equivalent to the section radius. For beams with non-circular cross-sections, the radius of the contact edge is equal to the radius of a circumscribed circle around the section. If

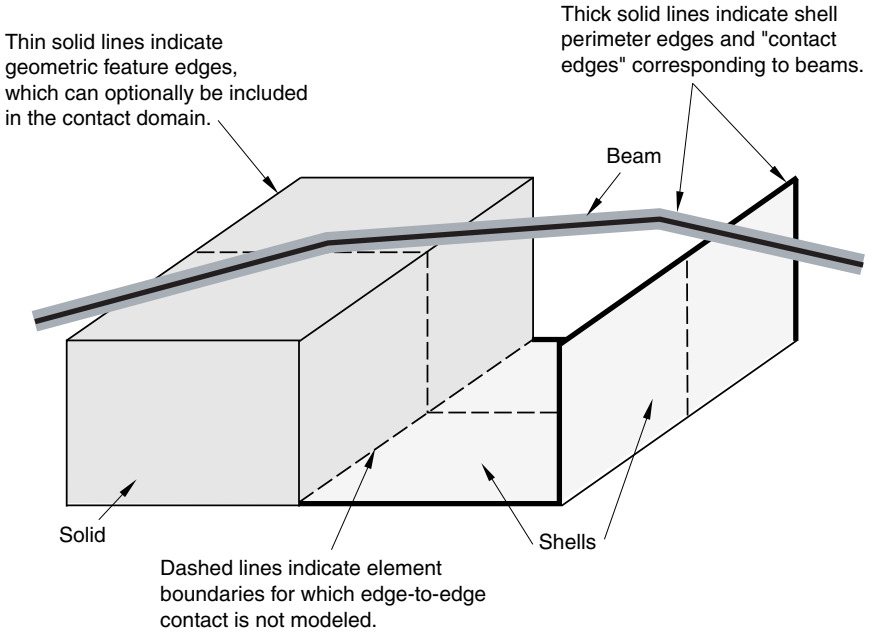


Figure 34.4.1–1 General contact domain, including edge-to-edge contact.

connected edges have different radii, a nodal radius is first computed as the minimum radius of the adjacent contact edges, and the radius of the edge cross-section is interpolated linearly over the length of the contact edge from the nodal values. Shell element edges reflect 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.

To model contact between edges that are not cylindrical in shape, surface elements can be attached to the edge nodes using surface-based tie constraints and node-to-face contact can be defined between the surface elements (see "Surface elements," Section 31.7.1). This technique is useful for modeling geometric details important to the contact definition that are not modeled with the underlying element geometry. Surface elements can also be defined around shell elements in which Abaqus has reduced the contact thickness (i.e., if the thickness exceeds the surface facet edge lengths or diagonal lengths) so that the true surface thickness can be modeled. However, using surface elements with general contact requires a physically reasonable mass to be associated with the surface element nodes, and care must be taken not to alter the bulk mass properties when transferring mass to the surface elements from the underlying elements.

By default, when a surface is used in a general contact interaction, all applicable facets, analytical rigid surfaces, nodes, perimeter 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," Section 34.4.2. Geometric feature edges and

perimeter edges do not have to be included explicitly in a surface definition (by using edge identifiers) for them to be considered for edge-to-edge contact.

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," Section 2.3.5). 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" in "Eulerian analysis," Section 14.1.1, for more information.

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 appear only once per step.

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.

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.

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. Only one contact inclusions definition and one contact exclusions definition are allowed per step.

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," Section 34.4.2). 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," Section 31.5.3, 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; any data lines specified will be ignored.

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 may contain multiple forming tools, but not all of 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.

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 master-slave contact surface weighting for a particular general contact surface pair, contact exclusions are generated automatically for the master-slave orientation

opposite to that specified (see "Contact formulation for general contact in Abaqus/Explicit," Section 36.2.1, 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. If an appropriate "interior" surface is defined, the surface topology will evolve to match the exterior of elements that have not failed. 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," Section 2.3.3).

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," Section 22.2.8). 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 34.4.1–2 for an example of an eroding solid).

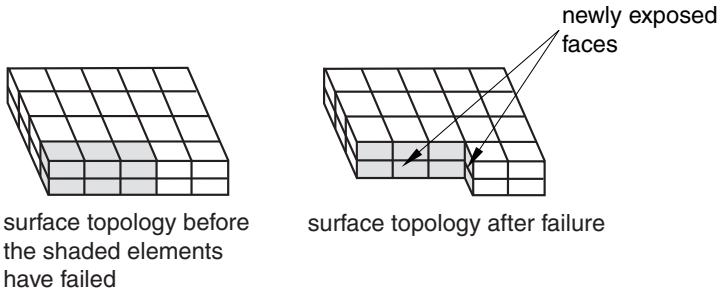


Figure 34.4.1–2 Topology of an eroding contact surface.

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 (i.e., 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," Section 34.4.5.

Erosion of surfaces specified on solid elements

For a solid element mesh consisting of elements that may fail, every face that can potentially be involved in contact (both exterior and interior faces) should be included in the contact domain. The general contact algorithm will activate and deactivate faces as necessary when elements fail.

For example, you define an element set **ELERODE** that contains all the solid elements in the model that refer to a material failure model. First, you must create a surface **SURFERODE** containing all of the interior and exterior faces of these elements. You could define this surface using the automatic free surface and interior surface generation methods in Abaqus/Explicit. Assuming all the elements in **ELERODE** are of type C3D8R, you could alternatively define the surface by specifying the faces S1 through S6 directly. See "Creating surfaces on solid, continuum shell, and cohesive elements" in "Element-based surface definition," Section 2.3.2, for a discussion of these three methods.

Next, you must construct 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 34.4.1–1.

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**.

Table 34.4.1–1 Contact inclusions definitions.

Contact inclusions	Input file syntax	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**	`, SURFERODE`	First Surface: (All*) Second Surface: SURFERODE
Self-contact for **SURFERODE** specifies self-contact between the eroding bodies	`SURFERODE,`	First Surface: SURFERODE Second Surface: (Self)

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 of the surrounding faces have failed. A perimeter edge (e.g., 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.

Memory use

The amount of contact data used to describe the surface topology is proportional to the number of faces included in the contact domain. Including a large number of interior faces in the contact domain can potentially increase memory use significantly compared to analyses in which the contact domain is defined using only exterior faces. Consider creating a surface on a cubic mesh of C3D8R elements with n elements per side. A surface including the exterior faces of the mesh (suitable for modeling contact without element failure) would contain $6n^2$ element faces. A surface including both exterior and interior faces of the mesh (suitable for modeling contact with element failure for every element in the mesh) would contain $6n^3$ element faces. For large meshes the memory use can increase easily by an order of magnitude when interior element faces are included in the contact domain to model erosion. Therefore, it is recommended to include only those interior element faces in the contact domain that could possibly participate in contact.

Output

The surfaces that compose the general contact domain are available as output in addition to the contact analysis output variables.

General contact domain surfaces

Abaqus/Explicit generates the following internal surfaces when a general contact domain is defined: **General_Contact_Faces_*k***, **General_Contact_Edges_*k***, and **General_Contact_Nodes_*k***, where ***k*** is the step number. **General_Contact_Nodes_*k*** contains only nodes in the general contact domain that are not included in the other two surfaces. For example, **General_Contact_Faces_2** 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. These internal surfaces can be viewed using display groups in the Visualization module of Abaqus/CAE (see the Abaqus/CAE User's Manual). 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 "Surface output in Abaqus/Standard and Abaqus/Explicit" in "Output to the output database," Section 4.1.3, for more information). The available variables are contact pressure, normal contact force, frictional force, and whole surface resultant quantities (i.e., 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 variable CPRESS (contact pressure) 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) 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).

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 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.

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.

34.4.2 ASSIGNING SURFACE PROPERTIES FOR GENERAL CONTACT IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Defining general contact interactions in Abaqus/Explicit," Section 34.4.1
- *CONTACT
- *SURFACE PROPERTY ASSIGNMENT
- "Specifying surface property assignments for general contact," Section 15.13.4 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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 applied selectively to particular regions within a general contact domain; and
- cannot be applied to analytical rigid surfaces.

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.

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) is appropriate for most analyses; one exception is sheet forming analysis, in which the thinning of a sheet significantly influences contact. This case can be modeled by specifying that the decreasing parent element thickness should be used. As a third alternative, you can specify a value for the surface thickness. A nonzero thickness can be assigned to solid element surfaces, for example, to model the effect of a finite-thickness surface coating. "Element-based surface definition," Section 2.3.2, contains information on the spatial variation of the surface thickness.

Specifying the original or decreasing thickness 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 will scale back the contact thickness automatically where necessary without affecting the thickness used in the element computations for the underlying elements. Diagnostic information is provided in the status (`.sta`) file if such scaling is performed.

To bypass this limitation on thickness, the contact surface can be modeled with surface elements (see "Surface elements," Section 31.7.1). The surface elements must be attached to the underlying elements using a surface-based tie constraint (see "Mesh tie constraints," Section 33.3.1), and a physically reasonable mass must be associated 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, contact controls settings can be used to limit the thickness reduction checks (see "Contact controls for general contact in Abaqus/Explicit," Section 34.4.5).

The "bull-nose" effect that occurs at shell perimeters with the contact pair algorithm (see "Assigning surface properties for contact pairs in Abaqus/Explicit," Section 34.5.2) 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. Contact controls settings can be used to turn off the bull-nose prevention checks (see "Contact controls for general contact in Abaqus/Explicit," Section 34.4.5).

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 34.4.2–1 and Table 34.4.2–1). 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. 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," Section 2.1.3), the nodal surface thickness may not correspond exactly to the specified nodal thickness (see node 4 in Figure 34.4.2–2 and Table 34.4.2–2).

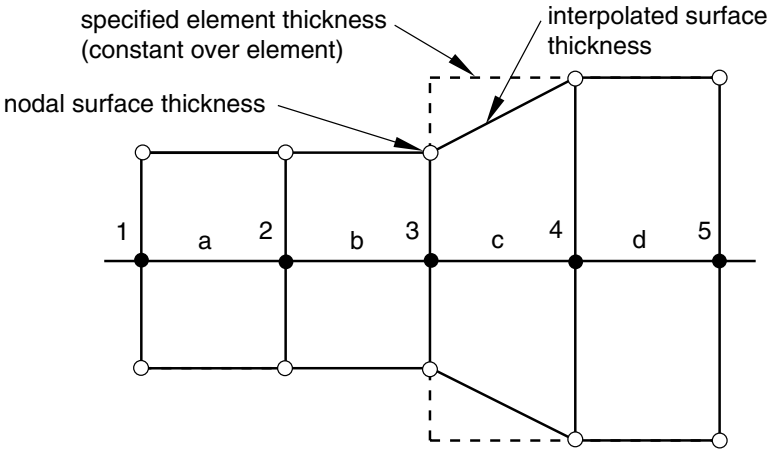


Figure 34.4.2–1 Continuous variation of surface thickness across facet boundaries.

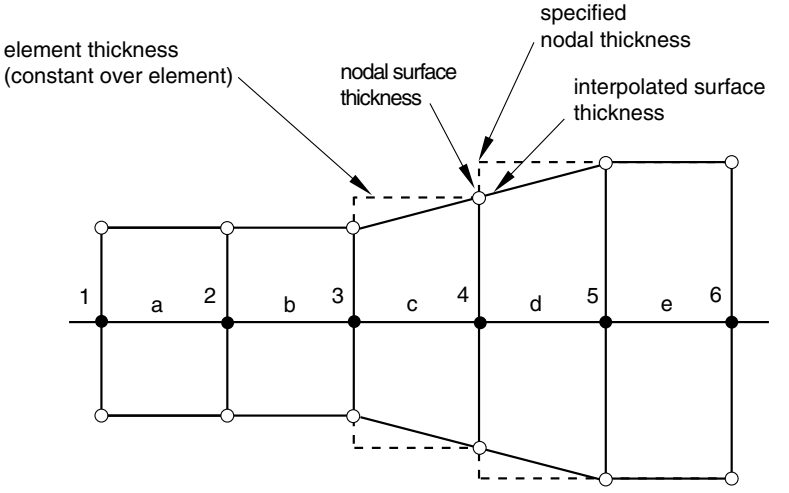


Figure 34.4.2–2 Small discrepancy between the nodal surface thickness and the specified nodal thickness.

Table 34.4.2–1 Thicknesses corresponding to Figure 34.4.2–1.

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

Table 34.4.2–2 Thicknesses corresponding to Figure 34.4.2–2.

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

Node	Element	Specified nodal thickness	Element thickness (average of specified nodal thickness)	Nodal surface thickness (minimum of adjacent element thicknesses)
	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, ORIGINAL (default)

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

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

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, THINNING

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

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Shell/Membrane thickness assignments: Edit**: Select surface, click the arrows to transfer surface 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, *value*

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

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Shell/Membrane thickness assignments: Edit**: Select surface, click the arrows to transfer surface 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 "Controlling initial contact status for general contact in Abaqus/Explicit," Section 34.4.4). 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, *value or label*, *scale_factor*

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

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Shell/Membrane thickness assignments: Edit**: Select surface, click the arrows to transfer surface 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.

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. At complex intersections (edges connected to more than two faces) the surface offset is set to zero. Figure 34.4.2–3 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.

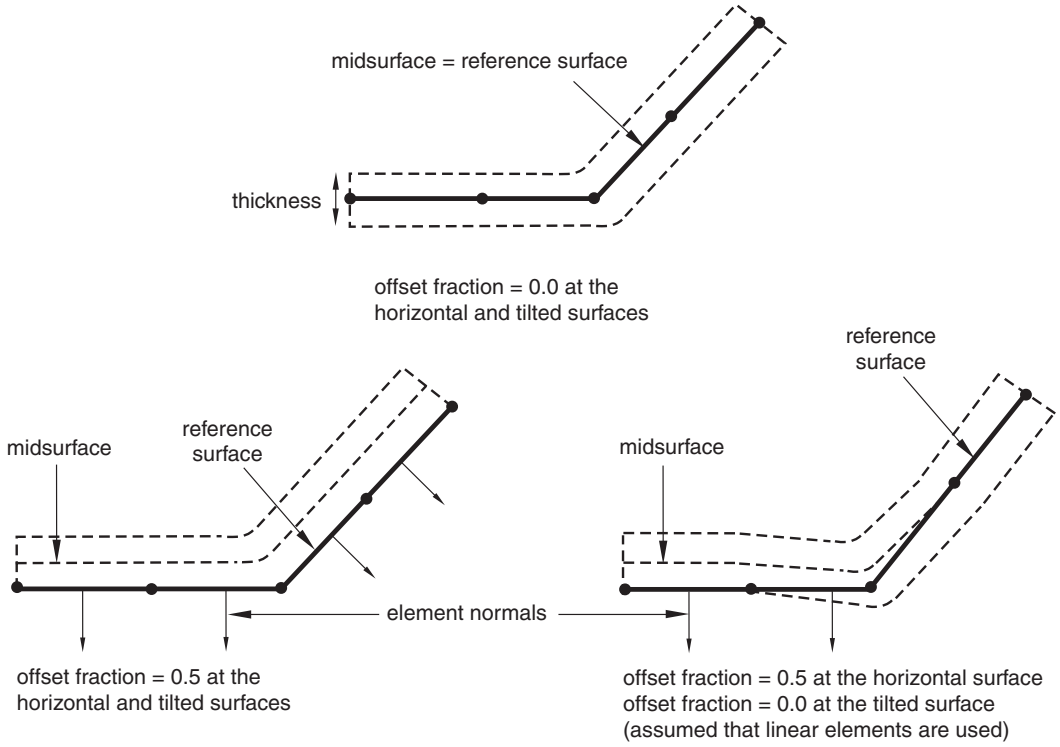


Figure 34.4.2–3 Specifying surface offsets for general contact.

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, ORIGINAL

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

*SURFACE PROPERTY ASSIGNMENT, PROPERTY=OFFSET FRACTION
surface, *offset*

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.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Shell/Membrane offset assignments: Edit**: Select surface, and click the arrows to transfer surface 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. By default, edge-to-edge contact in the general contact algorithm in Abaqus/Explicit accounts for perimeter edges as well as "contact edges" of beam and truss elements.

You can control which feature edges should be activated in the general contact domain by specifying feature edge criteria. By default, only perimeter edges are activated. Feature edge criteria have no effect on "edges" of beam and truss elements—they are activated by their inclusion in the contact domain.

The feature angle

The feature angle is the angle formed between the normals of the two facets connected to an edge. The angles between facets are based on the initial configuration. A negative angle will result at concave meetings of facets; therefore, these edges are never included in the contact domain. Figure 34.4.2–4 shows some examples of how the feature angle is calculated for different edges.The feature angle for edge A is 90° (the angle between n_1 and n_2); the feature angle for edge B is −25° (the angle between n_2 and n_3). Edge C forms a T-intersection with three facets (shown in two dimensions in Figure 34.4.2–5); its feature angles are 0°, −90°, and −90°. Perimeter edges (for example, edge D in Figure 34.4.2–4) 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 34.4.2–6 illustrates further how the feature angle is used to determine which geometric feature edges should be activated in the general contact domain. 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.

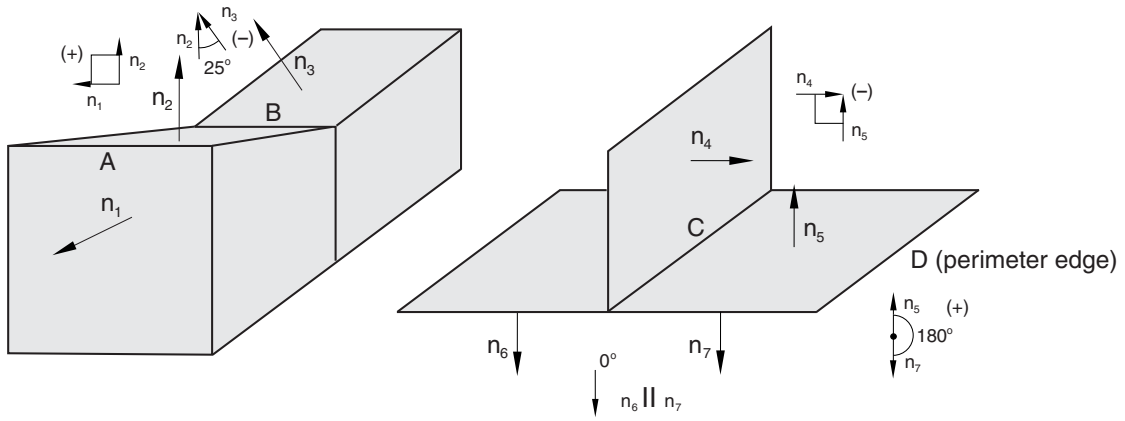


Figure 34.4.2–4 Calculating the feature angle.

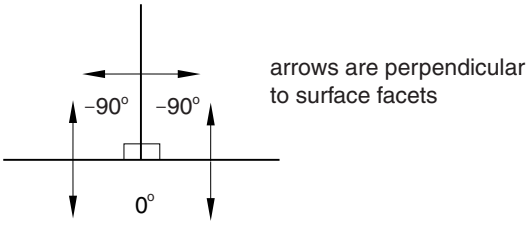


Figure 34.4.2–5 Feature angles for a T-intersection (for example, edge C in Figure 34.4.2–4).

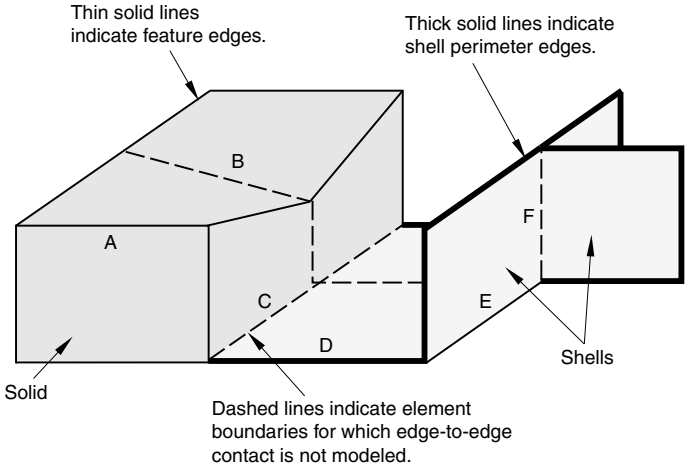
Specifying that only perimeter edges should be activated

By default, only perimeter edges are included in the general contact domain. 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 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, PERIMETER EDGES (default)

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

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**: Select surface, click the arrows to transfer surface to list of feature assignments, and enter PERIMETER in the **Feature Edge Criteria** column.



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 34.4.2–6 Feature edges activated in the general contact domain for a cutoff feature angle of 20°.

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," Section 2.3.2) is used to specify the edges to activate.

Input File Usage: *SURFACE PROPERTY ASSIGNMENT, PROPERTY=FEATURE
EDGE CRITERIA
surface, PICKED EDGES

Abaqus/CAE Usage: Specifying particular feature edges to be activated is not supported in Abaqus/CAE.

Specifying that all feature edges should be activated

You can choose to activate all edges in a given surface in the general contact domain. This will activate all edges of every face specified in the given surface.

Input File Usage: *SURFACE PROPERTY ASSIGNMENT, PROPERTY=FEATURE
EDGE CRITERIA
surface, ALL EDGES

Abaqus/CAE Usage: Specifying that all feature edges should be activated is not supported in Abaqus/CAE.

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, NO FEATURE EDGES

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

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**: Select surface, click the arrows to transfer surface to list of feature assignments, and enter NONE in the **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. The cutoff feature angle cannot be set to less than 20°. Significant edge-to-edge contact can be enforced for cutoff feature angles of 20° without negatively affecting performance; allowing a cutoff feature angle of less than 20° could severely degrade performance and would not affect the analysis results significantly compared to a cutoff angle of 20°. As described previously, you can activate additional feature edges if needed.

Input File Usage: *SURFACE PROPERTY ASSIGNMENT, PROPERTY=FEATURE EDGE CRITERIA
surface, *feature_angle_value*

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

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**: **Surface Properties**: **Feature edge criteria assignments: Edit**: Select surface, click the arrows to transfer surface to list of feature assignments, and enter a value for the cutoff feature angle (in degrees) in the **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, the input shown in the following table 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:

Input File Syntax	Abaqus/CAE Syntax
surf1, NO FEATURE EDGES	Surface: **surf1**, Feature Edge Criteria: NONE
surf2, PERIMETER EDGES	Surface: **surf2**, Feature Edge Criteria: PERIMETER
surf3, 30	Surface: **surf3**, Feature Edge Criteria: 30

Primary and secondary feature edges

To cut down on the computational cost in certain situations, it may be desirable to identify a limited number of feature edges on a surface (presumably at locations where there are sharp gradients in the surface normals) as "primary" feature edges. A more relaxed criterion can be used to denote certain other edges on the surface as "secondary" feature edges. If secondary feature edges are specified in addition to primary feature edges, 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 34.4.2–3.

Table 34.4.2–3 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

Primary Feature Edge Criterion	Secondary Feature Edge Criterion
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, *primary feature edge criterion*, ALL REMAINING EDGES

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

Abaqus/CAE Usage: Secondary feature edges are not supported in Abaqus/CAE.

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, *primary feature edge criterion*, PICKED EDGES

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

Abaqus/CAE Usage: Secondary feature edges are not supported in Abaqus/CAE.

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, *primary feature edge criterion*, PERIMETER EDGES

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

Abaqus/CAE Usage: Secondary feature edges are not supported in Abaqus/CAE.

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, *primary feature edge criterion*, *feature_angle_value*

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

Abaqus/CAE Usage: Secondary feature edges are not supported in Abaqus/CAE.

Specifying that edges are activated only as secondary feature edges

For a particular surface you may 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, NO FEATURE EDGES, *secondary feature edge criterion*

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

Abaqus/CAE Usage: Secondary feature edges are not supported in Abaqus/CAE.

34.4.3 ASSIGNING CONTACT PROPERTIES FOR GENERAL CONTACT IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Defining general contact interactions in Abaqus/Explicit," Section 34.4.1
- "Mechanical contact properties: overview," Section 35.1.1
- "Contact pressure-overclosure relationships," Section 35.1.2
- "Contact damping," Section 35.1.3
- "Frictional behavior," Section 35.1.5
- ∗CONTACT
- ∗CONTACT PROPERTY ASSIGNMENT
- ∗SURFACE INTERACTION
- "Specifying and modifying contact property assignments for general contact," Section 15.13.2 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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.

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, *surface_2*, *interaction_property_name*

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 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 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.

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **General contact (Explicit)**:
Contact Properties:
Individual property assignments: Edit: select the surfaces 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
```

34.4.4 CONTROLLING INITIAL CONTACT STATUS FOR GENERAL CONTACT IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Defining general contact interactions in Abaqus/Explicit," Section 34.4.1
- *CONTACT
- *CONTACT CLEARANCE
- *CONTACT CLEARANCE ASSIGNMENT
- "Producing a deformed shape plot," Section 43.5 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Initial clearances for surface interactions included in the general contact domain:

- are set to zero automatically for small initial overclosures (e.g., for small penetrations caused by numerical roundoff when a graphical preprocessor such as Abaqus/CAE is used);
- can be specified to resolve large initial overclosures that are not resolved automatically;
- can be specified to separate entangled double-sided surfaces;
- can be specified to model an initial gap between surfaces;
- are enforced without creating any strains or momentum in the model;
- should not be specified to correct gross errors in the mesh design; and
- can be used to identify an initially bonded node set in crack propagation analyses.

Default adjustments for initial overclosures in the first step of the simulation

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. The adjustments are made with strain-free initial displacements. This automatic adjustment of surface position is intended to correct only minor mismatches associated with mesh generation and is done even when the interaction is defined through user subroutine **VUINTERACTION**.

Conflicting adjustments from separate contacts, boundary conditions, tie constraints, coupling constraints, and rigid body constraints can cause incomplete resolution of initial overclosures. This can occur, for example, when a slave node is pinched between two master facets. Initial overclosures that are not resolved by repositioning nodes are stored as temporary contact offsets to avoid large contact forces at the beginning of an analysis. The penalty contact force is computed as $f = k(d_{cur} - d_0)$; where k is the penalty stiffness, d_0 is the initial unresolved penetration distance, and d_{cur} is the current penetration distance. If d_{cur} ever decreases below d_0, d_0 is reset to d_{cur}.

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 slave node lies within the thickness of the underlying element, and the initial penetration will be resolved by moving the slave node to the nearest free surface as shown in Figure 34.4.4–1.

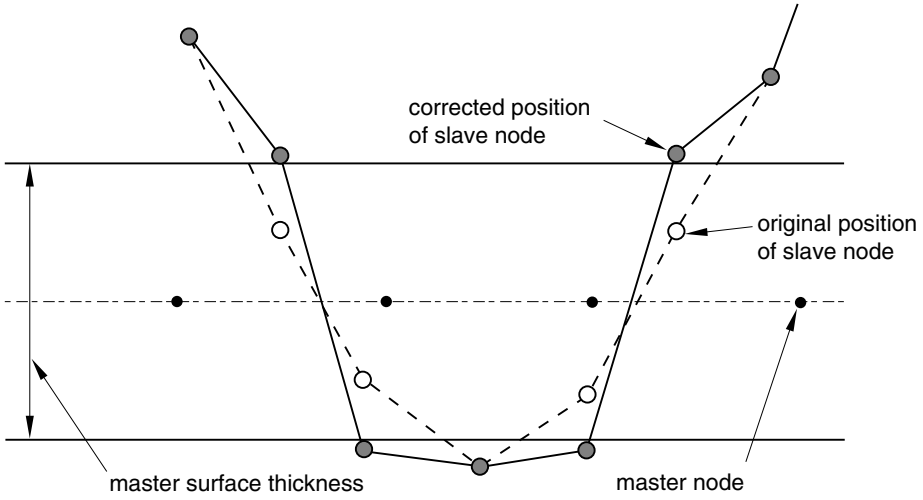


Figure 34.4.4–1 Correction of initial overclosure for contact involving two double-sided surfaces.

Slave nodes that are trapped on opposite sides of a double-sided master surface will often lead to serious problems, which may not become apparent until later in the analysis. Surfaces that are initially crossed often indicate a modeling problem for single-sided surfaces as well, because the initial search for slave nodes in the interior of solids is limited to a distance of about 15% of the facet dimensions; slave nodes more deeply penetrated than this are ignored by the algorithm to adjust initial overclosures.

Initial overclosure information—including node adjustment data, contact offsets, crossed surfaces, nodes that could not be corrected, and any warnings—is written to the status (`.sta`) file, the message (`.msg`) file, and the output database (`.odb`) file. The default tolerance used to report gross initial penetrations, which could indicate an error with your model definition, depends on the contact type. Node-to-surface contact uses the characteristic length of the contact facet, edge-to-edge contact uses the length of the tracked edge, and the typical element dimension is used for node-to-analytical rigid surface contact. For more information on the overclosure warnings, see "Contact diagnositcs in an Abaqus/Explicit analysis," Section 37.2.1, and Chapter 41, "Viewing diagnostic output," of the Abaqus/CAE User's Manual.

Default adjustments of overclosed surfaces during subsequent steps in the simulation

Initial penetrations are stored as temporary contact offsets that do not generate contact forces in the following cases:

- If the general contact domain is created in steps other than the first step (i.e., the contact definition follows a step in which no contact was defined) or
- if an Abaqus/Standard analysis is imported into Abaqus/Explicit and the contact interaction is not defined with user subroutine **VUINTERACTION**.

However, deep penetrations may not be treated correctly; they may be ignored or, in the case of penetrations past the midsurface of shells, the wrong contact directions may be used. Initial overclosure and crossed surface diagnostics can be requested to diagnose these problems (see "Contact diagnositcs in an Abaqus/Explicit analysis," Section 37.2.1).

If the general contact domain is extended after the first step, Abaqus/Explicit does not take any special actions to gradually resolve initial penetrations for the newly introduced interactions: penalty contact forces will be applied proportional to the penetration, or the penetration may be ignored. In addition, initial overclosure and crossed surface diagnostics are not available for these new interactions.

Specifying initial clearances and controlling initial overclosure adjustments

In some cases the default algorithm will not correctly resolve initial overclosures, or a precise initial gap (i.e., a positive clearance) between surfaces may need to be modeled. Specifically, deep penetrations may be ignored, tangled double-sided surfaces may not be separated correctly (see Figure 34.4.4–1), and gaps between curved surfaces in the discretized model may be inconsistent with the non-discretized model. To resolve these issues, you can define contact clearances and assign them to contact interactions. Examples are given below.

Defining contact clearances

You must assign a name to each contact clearance definition that is used to associate the clearance definition with a contact interaction.

Input File Usage: *CONTACT CLEARANCE, NAME=*clearance_name*

Abaqus/CAE Usage: Contact clearances for general contact are not supported in Abaqus/CAE.

Applying contact clearances by adjusting the nodal coordinates or by creating contact offsets

Clearances are applied to the model by adjusting the nodal coordinates or by creating contact offsets. By default, contact clearances are resolved by adjusting the nodal coordinates without creating strain or momentum in the model (this method can be used only in the first step of an analysis). Alternatively, contact offsets can be created for clearance specifications. These offsets are permanent (as opposed to temporary offsets created during the default initial overclosure resolution procedure) and are not ramped to zero as the surfaces separate. Contact offsets will also be created for clearances specified via nodal adjustments if the clearance violations cannot be resolved due to conflicting adjustments from separate contacts, boundary conditions, tie constraints, coupling constraints or rigid body constraints. Clearances can be applied via contact offsets in steps in which the whole contact domain is newly defined (i.e., no contact was defined in the previous step) and in the first step of an import analysis.

Input File Usage: Use the following option to apply contact clearances by adjusting the nodal coordinates (default):

*CONTACT CLEARANCE, NAME=*clearance_name*, ADJUST=YES

Use the following option to apply contact clearances by creating contact offsets:

*CONTACT CLEARANCE, NAME=*clearance_name*, ADJUST=NO

Abaqus/CAE Usage: Contact clearances for general contact are not supported in Abaqus/CAE.

Setting the value of the initial clearance

You can define the clearance as a single value for the whole interaction or as a nodal distribution to define a clearance per slave node (see "Distribution definition," Section 2.8.1). If a distribution is defined and the clearance is omitted for a slave node, the clearance value will be interpolated from the values at the master nodes. The slave node will be ignored if clearance values are specified for neither the slave node nor all of the nodes of the nearest master face.

The clearance values must be non-negative for slave nodes on solid element surfaces. The default value is 0.0 if a value or distribution is not given.

Input File Usage: *CONTACT CLEARANCE, NAME=*clearance_name*,
CLEARANCE=*value* or *distribution_name*

Abaqus/CAE Usage: Contact clearances for general contact are not supported in Abaqus/CAE.

Defining search zones

You can specify search distances to define "zones" above and below the surfaces. Slave nodes that lie within these zones will be given the specified clearance values with respect to their closest master faces by pulling them closer or pushing them farther away, regardless of their initial positions (overclosure or initial gap bigger than the clearance defined). Nodes whose closest point is a perimeter edge will be excluded from the clearance specification.

The default value for each search distance for solid elements is approximately one-tenth of the element size of the elements attached to the slave node. The default value for each search distance for structural elements (e.g., shell elements) is the thickness associated with the slave node.

Input File Usage: *CONTACT CLEARANCE, NAME=*clearance_name*,
SEARCH ABOVE=*value*, SEARCH BELOW=*value*

Abaqus/CAE Usage: Contact clearances for general contact are not supported in Abaqus/CAE.

Defining a search node set

As an alternative to specifying search distances, you can specify a search node set, containing the slave nodes for which clearance has been defined. Slave nodes that belong to this node set will be given the specified clearance values with respect to their closest master faces by pulling them closer or pushing them farther away, regardless of their initial positions (overclosure or initial gap bigger than the clearance defined). If a search node set has been specified, no clearance will be applied to slave nodes that do not belong to the specified search node set.

When a search node set is specified, there is a default search distance value associated with the maximum element size for solid elements or the thickness for structural elements (e.g., shell elements) associated with the nodes. The position of any node beyond the search distance is not adjusted.

Input File Usage: *CONTACT CLEARANCE, NAME=*clearance_name*,

SEARCH NSET=*node set name*

Abaqus/CAE Usage: Contact clearances for general contact are not supported in Abaqus/CAE.

Assigning contact clearances to contact interactions

You can assign initial clearance definitions to node-to-face interactions (except self-contact interactions) in the general contact domain. Initial clearance definitions cannot be assigned to node-to-analytical rigid surface interactions. For node-to-face interactions, the clearances defined between two surfaces apply to the interaction between the slave nodes in each surface and the whole of the other surface. When nodal adjustments are used to resolve clearance violations, the adjustments are made to satisfy the clearance specification with respect to each slave node's nearest master face in the initial configuration. Contact offsets are set to the value of the clearance violation between each slave node and its nearest master face in the initial configuration, and the slave nodes are then offset by that value with respect to the whole of the other surface during the analysis.

The surfaces specified must be single-sided and cannot contain complex intersections of faces (i.e., an edge cannot be connected to more than two faces) or discontinuous normals. Surfaces defined on solid elements will satisfy these requirements automatically. These restrictions arise from the definition of a clearance for surfaces on double-sided elements: a node has a positive (negative) clearance with respect to a surface if it is above (below) the surface as defined by the surface normal (see Figure 34.4.4–2). A negative clearance of a node with respect to a surface on double-sided elements does not indicate a state of penetration, but rather that the node has a gap with the other side of the elements underlying the surface.

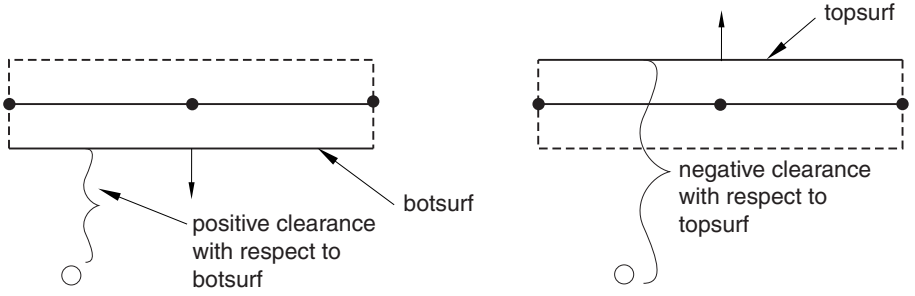


Figure 34.4.4–2 Contact clearance sign convention for double-sided elements.

By default, clearances are applied to all master-slave views of the surface pair that exist in the contact domain. In addition, if clearances between two element-based surfaces are specified to be resolved via nodal adjustments, the nodal adjustment procedure can be directed to perform the adjustments for one master-slave view of the surface pair (this applies only to the nodal adjustment procedure and does not apply to the contact formulation used between the surfaces during the analysis).

Input File Usage: Use the following option to specify clearances for all master-slave views of the given surface pair (default):

∗CONTACT CLEARANCE ASSIGNMENT

surface_1, *surface_2*, *clearance_name*

Use the following option to specify clearances between the nodes of the second surface and the faces of the first surface (the first surface is treated as the master surface):

∗CONTACT CLEARANCE ASSIGNMENT
surface_1, *surface_2*, *clearance_name*, MASTER

Use the following option to specify clearances between the nodes of the first surface and the faces of the second surface (the first surface is treated as the slave surface):

∗CONTACT CLEARANCE ASSIGNMENT
surface_1, *surface_2*, *clearance_name*, SLAVE

Abaqus/CAE Usage: Contact clearances for general contact are not supported in Abaqus/CAE.

Examples

The default algorithm to resolve initial overclosures does not detect penetrations of solid element surfaces that are greater than approximately 15% of the dimension of facets attached to the slave node. Figure 34.4.4–3 shows two solid elements with large initial penetrations that will not be detected during the default initial overclosure resolution procedure.

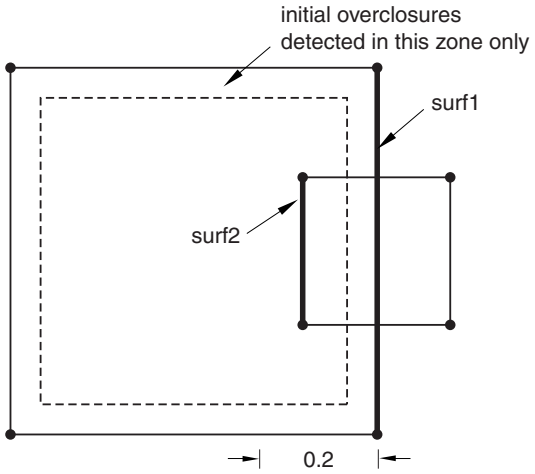


Figure 34.4.4–3 Undetected large penetrations of solid elements.

A zero clearance can be defined explicitly for the overclosed portions of this model to resolve the initial overclosures. Define the clearance definition as follows:

```
*CONTACT CLEARANCE, NAME=c1, ADJUST=YES, SEARCH BELOW=0.2
SEARCH ABOVE=0.0
```

and assign it to the interaction between **surf1** and **surf2**:

```
*CONTACT
*CONTACT CLEARANCE ASSIGNMENT
surf1, surf2, c1
```

The resulting adjustment is shown in Figure 34.4.4–4. Adjusting the nodal coordinates may degrade the mesh geometry by creating imperfections that were not initially present, may reduce the element size and correspondingly the stable time increment size, or may cause elements to invert and prevent the analysis from continuing. In such cases it is preferable to bypass the nodal coordinate adjustments and specify the storage of a contact offset.

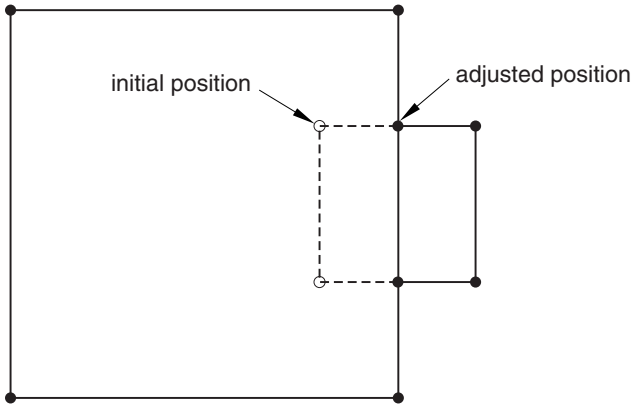


Figure 34.4.4–4 Resolution of large penetrations of solid elements.

The initial overclosure adjustment algorithm must also be directed to separate entangled double-sided surfaces. Figure 34.4.4–1 shows the default adjustments made for entangled shell surfaces assuming the nodes of **surf3** have fixed boundary conditions. Figure 34.4.4–5 shows the adjustments made from the following clearance definition and assignment:

```
*CONTACT CLEARANCE, NAME=c2, ADJUST=YES, SEARCH BELOW=1.5,
SEARCH ABOVE=0.0
...
*CONTACT
*CONTACT CLEARANCE ASSIGNMENT
surf3, surf4, c2
```

If the nodes of **surf3** are not fixed, the clearance interaction can be set to pure master-slave (with **surf3** defined as the master) to prevent the geometry of **surf3** from being modified.

In cases where the geometry of the mesh is important or if nodal adjustments conflict, contact offsets should be created. Conflicting nodal adjustments are a common problem when specifying clearances via nodal adjustment for curved surfaces with a balanced master-slave interaction. Adjustments of nodes

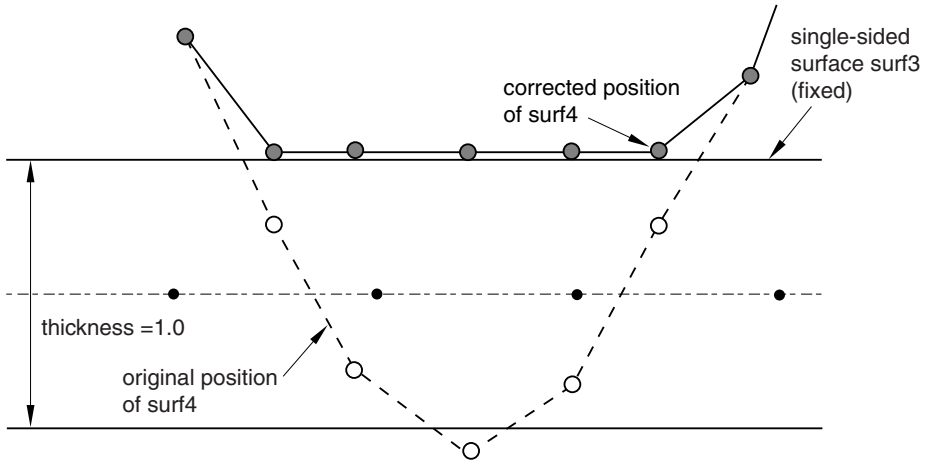


Figure 34.4.4–5 Separation of tangled double-sided surfaces.

tend to change the curvature of curved surfaces because the clearance "constraint" can be satisfied only if the surface meshes are coincident (and a zero clearance is specified) or if the surfaces are flat (see Figure 34.4.4–6).

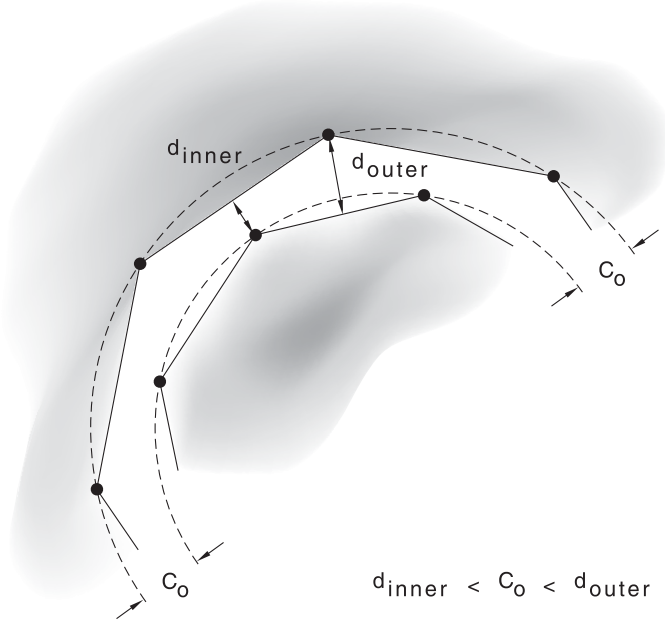


Figure 34.4.4–6 Specifying a uniform initial gap between concentric circular surfaces.

Identifying potentially partially bonded surfaces

You can specify a search node set to identify which slave nodes will be tagged as initially bonded in a VCCT crack propagation analysis. See “Crack propagation analysis,” Section 11.4.3, for more details.

Input File Usage: Use the following options:

∗CONTACT CLEARANCE, NAME=*clearance_name*,
SEARCH NSET=*node set name*
∗CONTACT CLEARANCE ASSIGNMENT
surface_1, *surface_2*, *clearance_name*

34.4.5 CONTACT CONTROLS FOR GENERAL CONTACT IN Abaqus/Explicit

Product: Abaqus/Explicit

References

- "Defining general contact interactions in Abaqus/Explicit," Section 34.4.1
- "Assigning surface properties for contact pairs in Abaqus/Explicit," Section 34.5.2
- *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 activate a nondefault tracking algorithm for node-to-face contact in particular regions within a general contact domain;
- 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," Section 36.2.3, 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," Section 1.1.11 of the Abaqus Benchmarks Manual, 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," Section 36.2.3, for further discussion).

The user-specified (variable) mass scaling does not take into account the effect of contact when it computes the necessary increase of mass. In general, this effect is not significant as the default penalty stiffness will decrease the stable time increment only by very small amounts. However, if high penalty scale factors are specified, the stable time increment could be reduced significantly despite the specified mass scaling.

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 nondefault tracking algorithm for node-to-face contact

A nondefault contact tracking algorithm is available that utilizes more local topological and geometric information in tracking contact between nodes and faces. This algorithm may lead to more robust contact tracking in certain modeling situations, for instance during the inflation event of a folded air-bag.

The tracking algorithm is activated on a surface-by-surface basis. You must specify the surface name for which the tracking algorithm needs to be activated. All contact interactions in the contact domain in which nodes of the specified surface contact faces belonging to either the surface itself (self-contact) or faces belonging to any other surface (for which node-to-face contact has not been excluded) will be tracked using the nondefault node-to-face tracking scheme.

The surface names used to specify the regions where the nondefault tracking algorithm should be used 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 nondefault tracking algorithm will be assigned to a subset of this domain. If the surfaces for which the nondefault tracking algorithm needs to be activated fall outside the general contact domain, the controls assignment is ignored.

Input File Usage: ∗CONTACT CONTROLS ASSIGNMENT, TYPE=FOLD TRACKING
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 nondefault tracking algorithm 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.

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.

Activating the default tracking algorithm for edge-to-edge contact

The default contact tracking algorithm utilizes more local information than the alternative tracking algorithm in tracking contact between edges and typically reduces the extent of global tracking required. The use of this algorithm may lead to smaller computational times in analyses that have extensive edge-to-edge contact defined (for example, during the inflation simulation of a folded airbag, where it may be desirable to activate all feature edges on the airbag membrane surface to accurately enforce contact during the inflation event).

The default tracking algorithm can be explicitly specified, though all edge-to-edge contact in the contact domain will be enforced using the default tracking algorithm if contact controls are not specified for the tracking algorithm.

Input File Usage: ∗CONTACT CONTROLS ASSIGNMENT, TYPE=ENHANCED
EDGE TRACKING (default)

This option must be used in conjunction with the ∗CONTACT option. This parameter setting applies to the entire general contact domain.

An alternative tracking algorithm for edge-to-edge contact

An alternative contact tracking algorithm is available that utilizes less local information than the default tracking algorithm in tracking contact between edges. This algorithm typically increases the extent of global tracking required and, hence, in most analyses the computational time. When the alternative edge

tracking algorithm is specified, all edge-to-edge contact in the contact domain is enforced using this algorithm.

Input File Usage: ∗CONTACT CONTROLS ASSIGNMENT, TYPE=EDGE TRACKING

If specified, this option must be used in conjunction with the ∗CONTACT option. This parameter setting applies to the entire general contact domain.

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 temporary 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.

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," Section 34.5.2). 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 "Defining general contact interactions in Abaqus/Explicit," Section 34.4.1) 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
```

34.5 Defining contact pairs in Abaqus/Explicit

- "Defining contact pairs in Abaqus/Explicit," Section 34.5.1
- "Assigning surface properties for contact pairs in Abaqus/Explicit," Section 34.5.2
- "Assigning contact properties for contact pairs in Abaqus/Explicit," Section 34.5.3
- "Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4
- "Contact controls for contact pairs in Abaqus/Explicit," Section 34.5.5

34.5.1 DEFINING CONTACT PAIRS IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Element-based surface definition," Section 2.3.2
- "Node-based surface definition," Section 2.3.3
- "Analytical rigid surface definition," Section 2.3.4
- "Contact interaction analysis: overview," Section 34.1.1
- *CONTACT CONTROLS
- *CONTACT PAIR
- *SURFACE
- "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining self-contact," Section 15.13.7 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Abaqus/Explicit provides two algorithms for modeling contact and interaction problems: the general contact algorithm and the contact pair algorithm. See "Contact interaction analysis: overview," Section 34.1.1, 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 (i.e., 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 have to use surfaces with matching meshes; and
- cannot be formed with one two-dimensional surface and one three-dimensional surface.

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," Section 34.5.2);
- the mechanical contact property models ("Assigning contact properties for contact pairs in Abaqus/Explicit," Section 34.5.3);
- the contact formulation ("Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2);
- the contact constraint enforcement method ("Contact constraint enforcement methods in Abaqus/Explicit," Section 36.2.3); and
- the algorithmic contact controls ("Common difficulties associated with contact modeling using contact pairs in Abaqus/Explicit," Section 37.2.2).

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" in "Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2, for details). See "Element-based surface definition," Section 2.3.2; "Node-based surface definition," Section 2.3.3; and "Analytical rigid surface definition," Section 2.3.4, 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 master-slave 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," Section 2.3.2); i.e., a zero surface thickness (see "Forcing zero surface thickness and offset" in "Assigning surface properties for contact pairs in Abaqus/Explicit," Section 34.5.2) 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" in "Assigning surface properties for contact pairs in Abaqus/Explicit," Section 34.5.2.
- 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 (i.e., 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 "Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4). 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 slave nodes in contact will be reinitialized. For example, a penalty contact slave node with a penetration past the midsurface of a double-sided master surface would be allowed to pass through the master 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," Section 2.3.2) 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," Section 22.2.8, for more information). Use general contact ("Defining general contact interactions in Abaqus/Explicit," Section 34.4.1) or node-based surfaces ("Node-based surface definition," Section 2.3.3) in such cases.
- The surface must be continuous, as discussed below.
- Continuum and structural elements cannot be mixed in the same surface definition.
- Deformable elements cannot be combined with elements that are part of a rigid body to define a single surface.

These restrictions do not apply to surfaces used with the general contact algorithm ("Defining general contact interactions in Abaqus/Explicit," Section 34.4.1).

The following restrictions apply to the surfaces forming a kinematic contact pair:

- Rigid surfaces must always be the master surface.
- Slave surfaces must be part of a deformable body.
- A node-based surface can be used only as a slave surface.

The following restrictions apply to the surfaces forming a penalty contact pair:

- Analytical rigid surfaces must always be the master surface.
- A node-based surface can be used only as a slave 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 master surface forming the contact pair should always point toward the slave surface. If, in the initial configuration of the model, a single-sided master surface's normal points away from its slave surface, Abaqus/Explicit will detect that the slave surface penetrates the master 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

"Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4). 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, Abaqus/Explicit, and Abaqus/CFD execution," Section 3.2.2) 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 34.5.1–1.

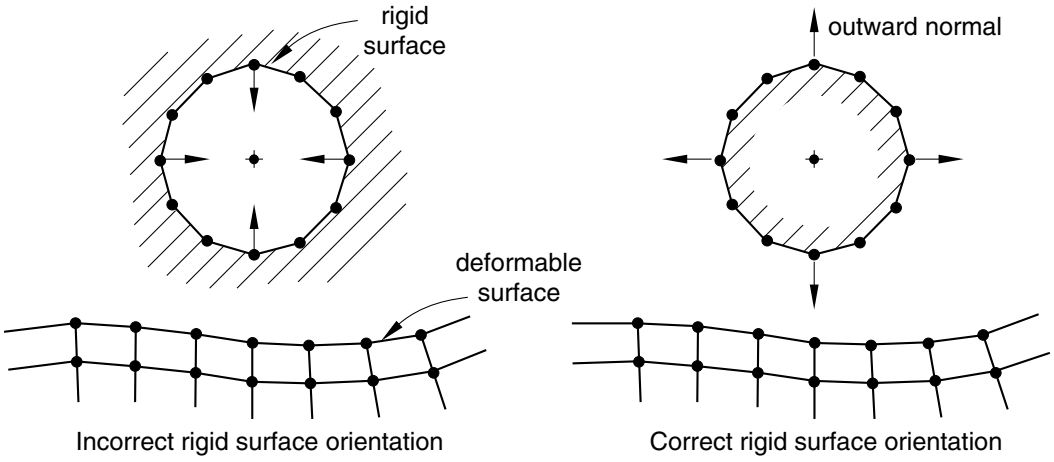


Figure 34.5.1–1 Example of proper and improper surface orientation with a rigid surface.

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," Section 2.3.2.

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 34.5.1–2 shows examples of valid and invalid two-dimensional surfaces for use in contact pairs.

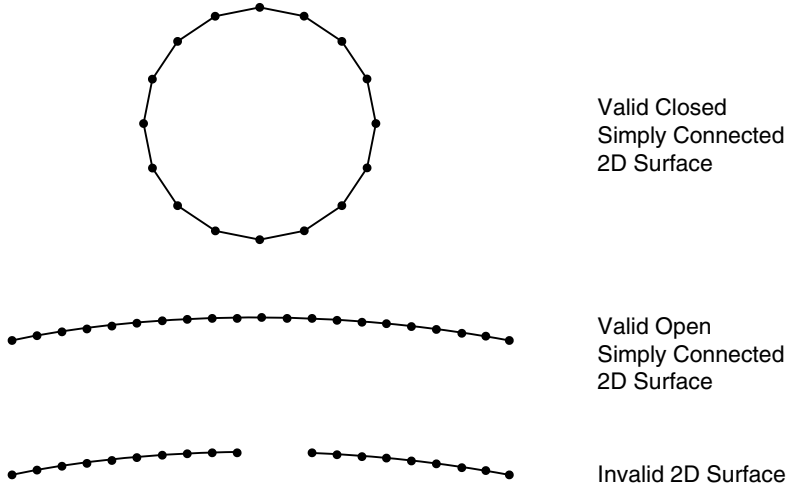


Figure 34.5.1–2 Valid and invalid 2-D surfaces.

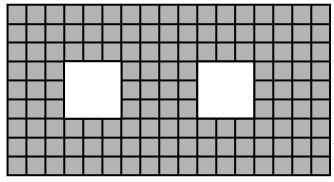
In three dimensions an edge of an element face belonging to a valid surface may be either on the perimeter of the surface or shared by one other face. Two element faces forming a contact pair surface cannot be joined just at a shared node; they must be joined across a common element edge. An element edge cannot be shared by more than two surface facets. Figure 34.5.1–3 illustrates valid and invalid three-dimensional surfaces for use in contact pairs.

The continuity requirement applies to both automatically generated free surfaces and surfaces defined with element face identifiers (see "Element-based surface definition," Section 2.3.2). Figure 34.5.1–4 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.

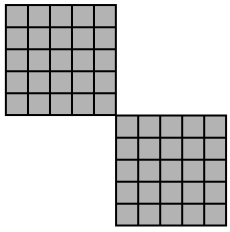
Restrictions for two-dimensional contact simulations

The following restrictions apply when defining a contact simulation for two-dimensional (planar) or axisymmetric problems:

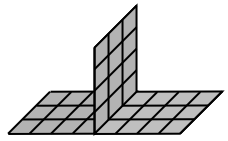
- A contact pair cannot involve a planar surface and an axisymmetric surface. This restriction applies only to deformable and element-based rigid surfaces.
- Defining a contact pair that contains two surfaces formed by planar elements of different sizes in the out-of-plane direction ("depth") is not 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



Valid Simply Connected Surface



Invalid Surface



Invalid Surface

Figure 34.5.1–3 Valid and invalid 3-D surfaces.

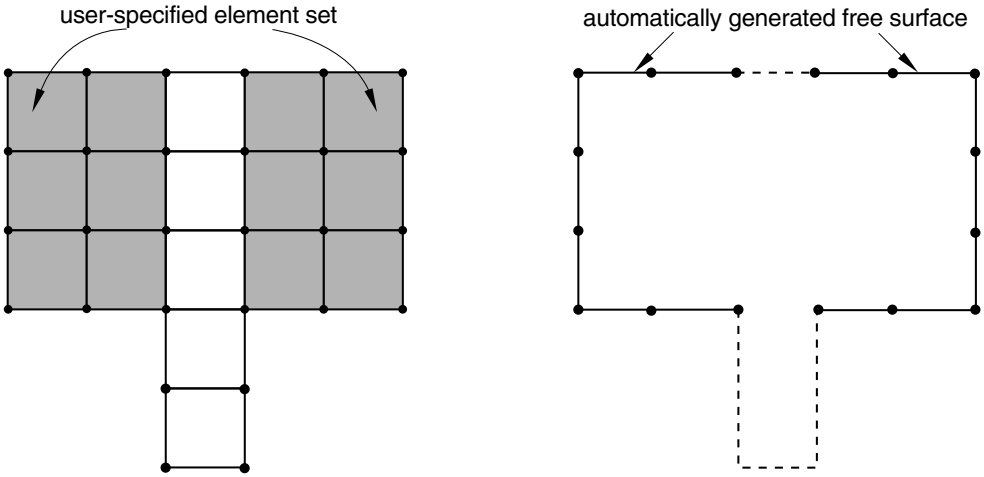


Figure 34.5.1–4 Automatic free surface generation.

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 slave surface in a pure master-slave 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 (i.e., 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 "Surface output in Abaqus/Standard and Abaqus/Explicit" in "Output to the output database," Section 4.1.3. Output from thermal interactions is discussed in "Thermal contact properties," Section 35.2.1.

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 (i.e., 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 (i.e., non-analytical) surface in a contact pair.

If FSLIPR is requested, FSLIPR (the magnitude of the slip rate for slave nodes in contact) can be contoured in Abaqus/CAE for each slave 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 slave surface in a contact pair if FSLIPR is requested. All of the slip rate variables associated with FSLIPR are zero whenever a slave node is not in contact.

If FSLIP is requested, FSLIPEQ (the length of the overall slip path for a slave node while it is in contact) can be contoured in Abaqus/CAE for each slave 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 slave 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 slave nodes are in contact.

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.

Detailed history output on the status of bonded surfaces is available from an Abaqus/Explicit simulation. Details can be found in "Breakable bonds," Section 35.1.9.

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

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

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.

34.5.2 ASSIGNING SURFACE PROPERTIES FOR CONTACT PAIRS IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Defining contact pairs in Abaqus/Explicit," Section 34.5.1
- *CONTACT PAIR
- *SURFACE
- "Specifying geometric properties for mechanical contact property options" in "Defining a contact interaction property," Section 15.14.1 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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," Section 29.3.1). 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 34.5.2–1 and Table 34.5.2–1). 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," Section 2.1.3), the nodal surface thickness may not correspond exactly to the specified nodal thickness (see node 4 in Figure 34.5.2–2 and Table 34.5.2–2). 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.

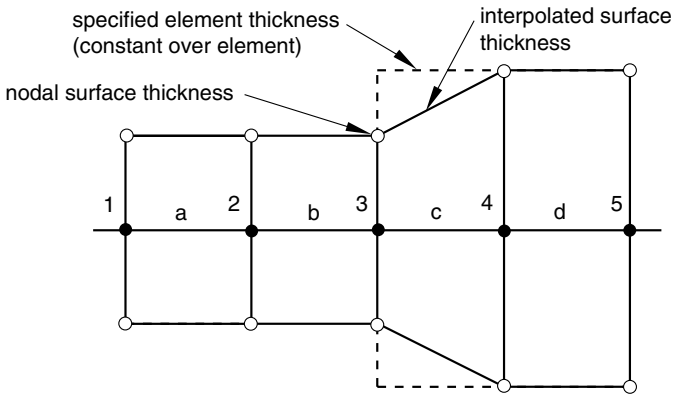


Figure 34.5.2–1 Continuous variation of surface thickness across facet boundaries.

Table 34.5.2–1 Thicknesses corresponding to Figure 34.5.2–1.

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

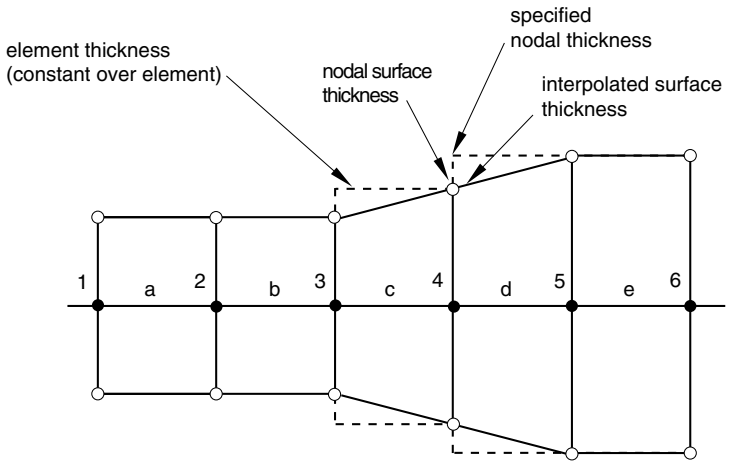


Figure 34.5.2–2 Small discrepancy between the nodal surface thickness and the specified nodal thickness.

Table 34.5.2–2 Thicknesses corresponding to Figure 34.5.2–2.

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 34.5.2–3 demonstrates this concept. Such "bull-nose" extensions are avoided when the general contact algorithm ("Defining general contact interactions in Abaqus/Explicit," Section 34.4.1) is used. The effect of a shell or rigid offset on a surface is shown in Figure 34.5.2–4. Poorly defined surfaces can result near corners if large offsets are present, as shown in Figure 34.5.2–5. 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).

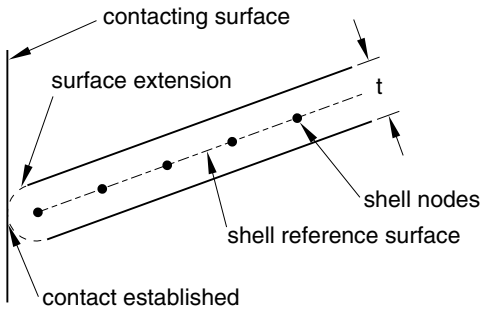


Figure 34.5.2–3 Extension of contact surface for edge contact without zero surface thickness.

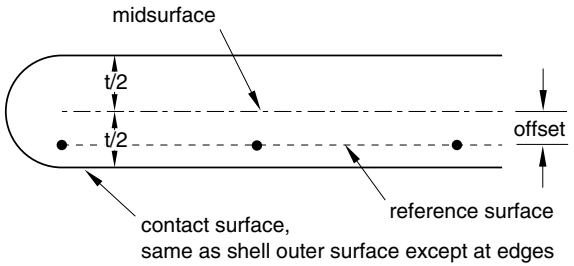


Figure 34.5.2–4 Extension of contact surface if a shell offset is present.

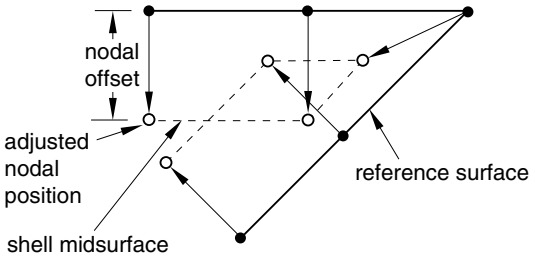


Figure 34.5.2–5 Example of a poorly defined surface near a corner when a large shell offset is present.

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 34.5.2–6).

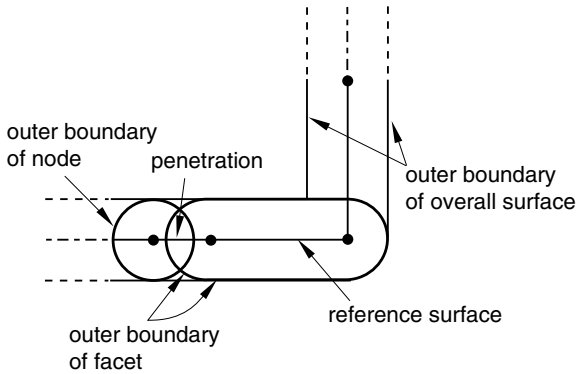


Figure 34.5.2–6 Undesired penetration resulting from a large thickness in a self-contact surface.

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 34.5.2–7).

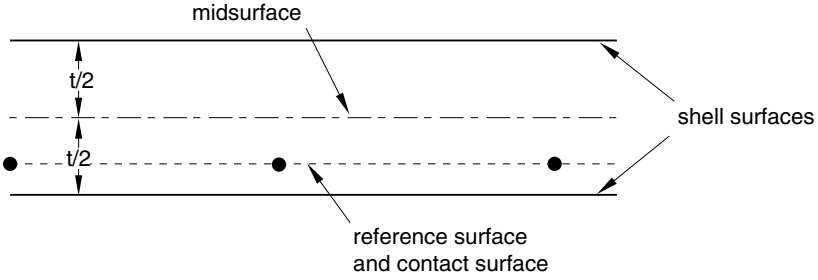


Figure 34.5.2–7 Contact surface with zero thickness and offset.

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 34.5.2–8.

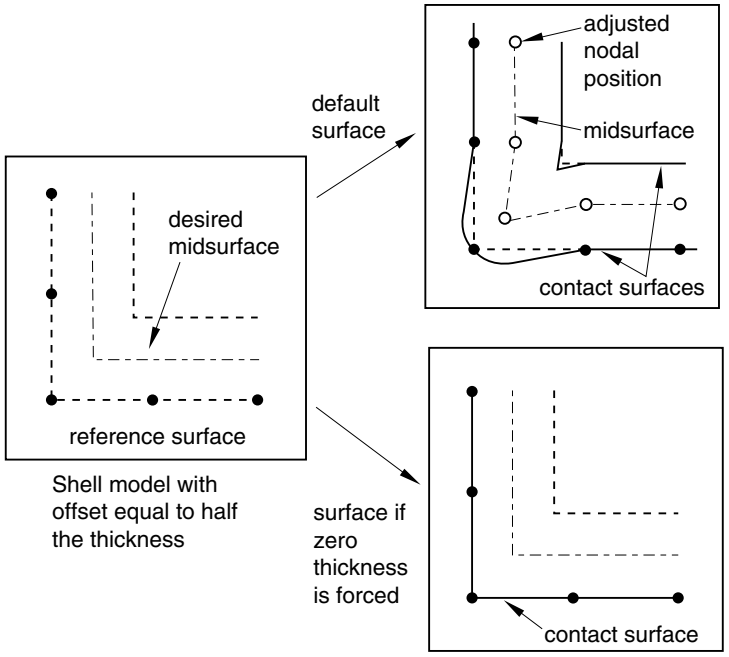


Figure 34.5.2–8 Forcing zero surface thickness when the shell offset is half the original shell thickness.

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 34.5.2–9). 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.

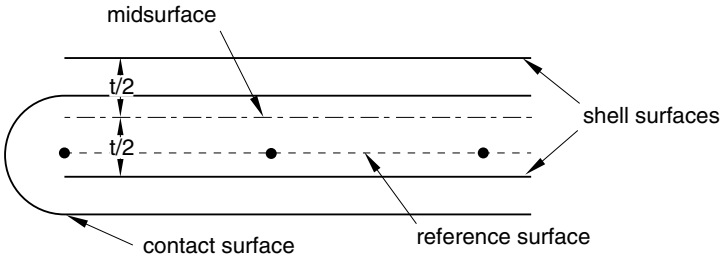


Figure 34.5.2–9 Contact surface with zero offset.

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" in "Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4). 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," Section 35.1.2) 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*

34.5.3 ASSIGNING CONTACT PROPERTIES FOR CONTACT PAIRS IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Mechanical contact properties: overview," Section 35.1.1
- "Contact pressure-overclosure relationships," Section 35.1.2
- "Contact damping," Section 35.1.3
- "Frictional behavior," Section 35.1.5
- "User-defined interfacial constitutive behavior," Section 35.1.6
- "Breakable bonds," Section 35.1.9
- *CONTACT PAIR
- *SURFACE INTERACTION
- "Interaction property editors," Section 15.9.3 of the Abaqus/CAE User's Manual

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 34.5.3–1 shows the mesh used in this example. For purposes of this example, a balanced master-slave contact pair is used. The property definition for the contact pair (**GRATING**) uses a friction model where μ=0.4.

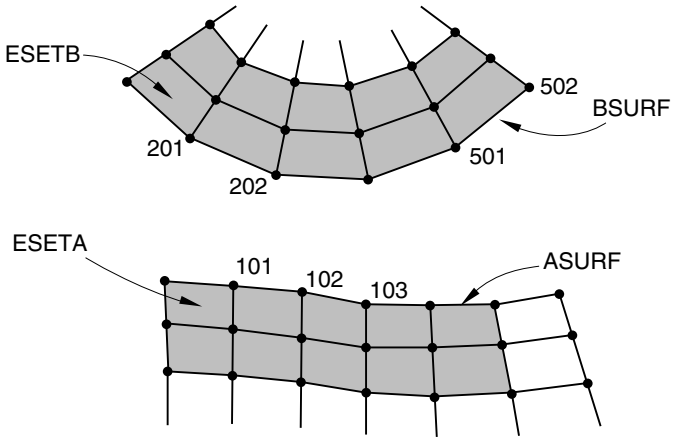


Figure 34.5.3–1 Surface interaction with friction.

```
*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
```

34.5.4 ADJUSTING INITIAL SURFACE POSITIONS AND SPECIFYING INITIAL CLEARANCES FOR CONTACT PAIRS IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Defining contact pairs in Abaqus/Explicit," Section 34.5.1
- *CLEARANCE
- *CONTACT PAIR
- "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Adjustments to the positions of the slave nodes in an Abaqus/Explicit contact pair:

- are performed for all contact pairs that have slave nodes that are overclosed and that do not have specified initial clearances, except when nodes of a rigid body act as slave 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 slave node that is pinched between two master surfaces.

If the small-sliding contact formulation (see "Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2) 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 slave nodes or user subroutine **VUINTER** is used. The adjustments are made with strain-free initial displacements to the slave nodes on the surfaces. Therefore, when a balanced master-slave 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 diagnositcs in an Abaqus/Explicit analysis," Section 37.2.1, for more information.

Some softened contact models have nonzero contact pressure at zero overclosure (see "Contact pressure-overclosure relationships," Section 35.1.2). 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 slave node is pinched between two master 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 slave node lies within the thickness of the underlying element, and the initial penetration will be resolved by moving the slave node to the nearest free surface as shown in Figure 34.5.4–1.

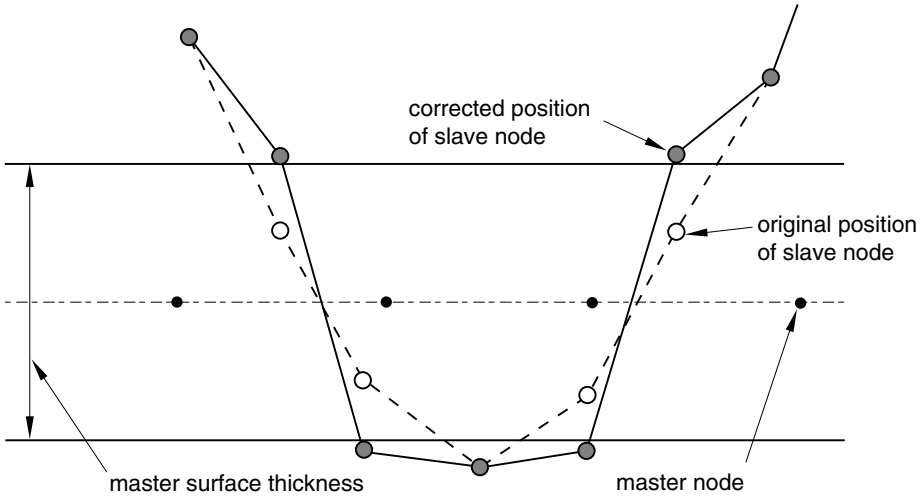


Figure 34.5.4–1 Correction of initial overclosure for a contact pair involving two double-sided surfaces.

A warning message will be issued to the status (`.sta`) file if two adjacent slave nodes (connected by a facet edge) are detected on opposite sides of a double-sided master 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 master surface, because adjacency cannot be determined among the node-based surface nodes. If the master surface is a single-sided surface, initial overclosures will be resolved using the surface normal of the master surface, as shown in Figure 34.5.4–2.

Having slave nodes trapped on opposite sides of a double-sided master 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, Abaqus/Explicit, and Abaqus/CFD execution," Section 3.2.2) is recommended prior to running a large contact pair analysis so that you can check for warning messages

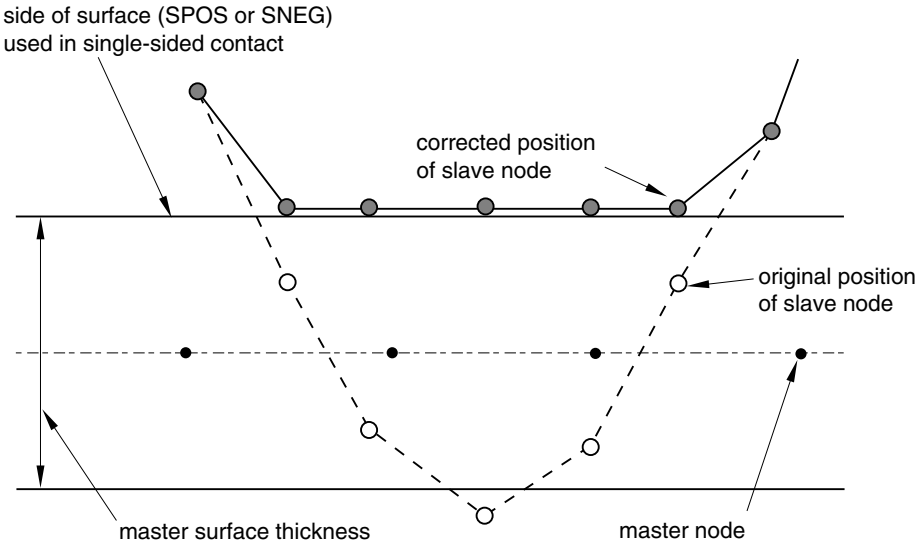


Figure 34.5.4–2 Correction of initial overclosure for a contact pair involving a single-sided and a double-sided surface.

in the status file (`.sta`) and check for mislocated adjacent slave nodes on opposite sides of the master 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 slave nodes only for penalty contact pairs. Initial penetrations of slave nodes that are part of a rigid body are not resolved with strain-free corrections; i.e., the slave 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 master-slave 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 master-slave relationship using a weighting factor; see "Contact surface weighting" in "Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2, for details.

Specifying initial clearance values precisely

You can define initial clearances and contact directions precisely for the nodes on the slave 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," Section 36.2.2).

The initial clearance value calculated at every slave node based on the coordinates of the slave node and the master surface is overwritten by the value that you specify. This procedure does not alter the coordinates of the slave nodes.

When the balanced-master slave 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 master 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 slave 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 slave surface. Any slave 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.

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," Section 36.2.2. When specifying spatially varying clearances, you can redefine the contact direction that Abaqus/Explicit uses with each slave node by specifying the components of this vector. The vector must define the global Cartesian components of the outward normal to the master surface.

Input File Usage: *CLEARANCE, SLAVE=*surface_name*, MASTER=*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 slave 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 slave 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*

34.5.5 CONTACT CONTROLS FOR CONTACT PAIRS IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Defining contact pairs in Abaqus/Explicit," Section 34.5.1
- *CONTACT CONTROLS
- "Specifying contact controls in an Abaqus/Explicit analysis," Section 15.13.9 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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," Section 36.2.3), 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," Section 1.1.11 of the Abaqus Benchmarks Manual, 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," Section 36.2.3, 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" in "Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2. 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 master faces near each slave 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 34.5.5–1 illustrates a situation that might require more frequent global tracking. The master surface is a valid surface, but it contains a hole. The slave node shown identifies the shaded element facet as the closest master surface facet during an increment. The local contact search looks at this master surface facet and its neighbors.

If the slave node displaces across the hole in relatively few increments, the potential contact between the slave node and the master 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 slave node moves rapidly across a deep valley in the master 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*

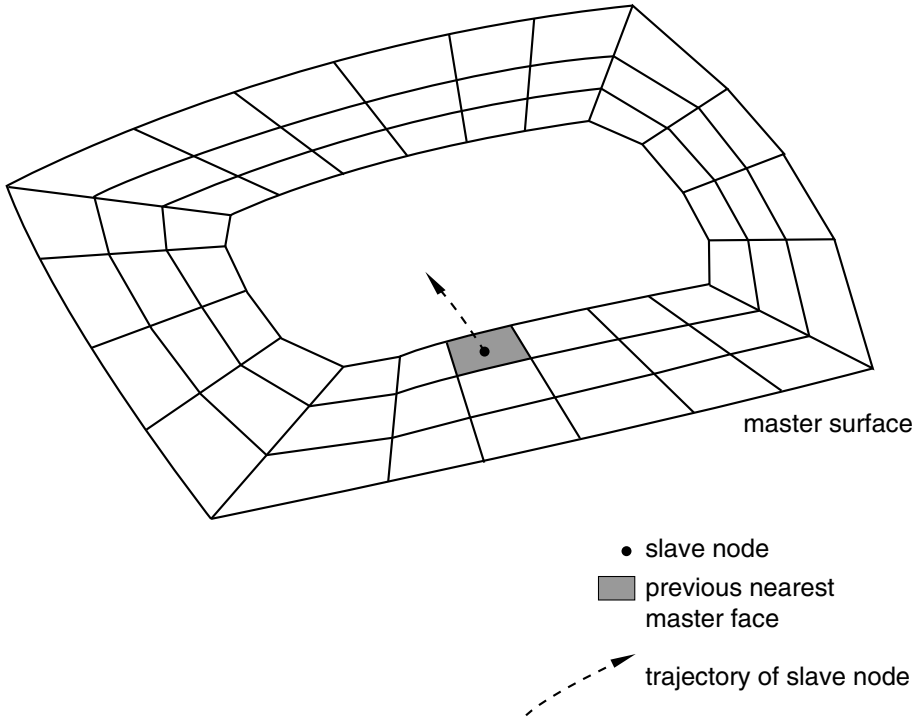


Figure 34.5.5–1 Example where local search may fail.

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 master 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 diagnositcs in an Abaqus/Explicit analysis," Section 37.2.1), and a more accurate algorithm is used to calculate each slave node's nearest point on the warped master 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*

35. Contact Property Models

Mechanical contact properties 35.1
Thermal contact properties 35.2
Electrical contact properties 35.3
Pore fluid contact properties 35.4

35.1 Mechanical contact properties

- "Mechanical contact properties: overview," Section 35.1.1
- "Contact pressure-overclosure relationships," Section 35.1.2
- "Contact damping," Section 35.1.3
- "Contact blockage," Section 35.1.4
- "Frictional behavior," Section 35.1.5
- "User-defined interfacial constitutive behavior," Section 35.1.6
- "Pressure penetration loading," Section 35.1.7
- "Interaction of debonded surfaces," Section 35.1.8
- "Breakable bonds," Section 35.1.9
- "Surface-based cohesive behavior," Section 35.1.10

35.1.1 MECHANICAL CONTACT PROPERTIES: OVERVIEW

References

- "Contact interaction analysis: overview," Section 34.1.1
- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- "Assigning contact properties for general contact in Abaqus/Explicit," Section 34.4.3
- "Assigning contact properties for contact pairs in Abaqus/Explicit," Section 34.5.3
- "Contact pressure-overclosure relationships," Section 35.1.2
- "Contact damping," Section 35.1.3
- "Contact blockage," Section 35.1.4
- "Frictional behavior," Section 35.1.5
- "User-defined interfacial constitutive behavior," Section 35.1.6
- "Pressure penetration loading," Section 35.1.7
- "Interaction of debonded surfaces," Section 35.1.8
- "Breakable bonds," Section 35.1.9
- "Surface-based cohesive behavior," Section 35.1.10
- *SURFACE INTERACTION
- "Understanding interaction properties," Section 15.4 of the Abaqus/CAE User's Manual

Overview

In a mechanical contact simulation the interaction between contacting bodies is defined by assigning a contact property model to a contact interaction (see "Defining contact pairs in Abaqus/Standard," Section 34.3.1; "Assigning contact properties for general contact in Abaqus/Explicit," Section 34.4.3; and "Assigning contact properties for contact pairs in Abaqus/Explicit," Section 34.5.3, 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 slave surface contact the master 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," Section 35.1.2, and the constraint methods available to enforce these relationships are discussed in "Contact constraint enforcement methods in Abaqus/Standard," Section 36.1.2.

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," Section 35.1.3.

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," Section 35.1.4.

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," Section 35.1.5.

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," Section 35.1.6.

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 "Pressure penetration loading," Section 35.1.7.

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," Section 11.4.3. The details of the contact interaction model after debonding are discussed in "Interaction of debonded surfaces," Section 35.1.8.

Defining breakable bonds in Abaqus/Explicit

In Abaqus/Explicit you can define breakable bonds that connect the interacting surfaces. The kinematic contact pair algorithm must be used when defining breakable bonds.

The breakable bonds affect both the relative tangential motion and the motion normal to the surfaces. Breakable bonds cannot be used with analytical rigid surfaces. The details of the breakable bond model, known as the spot weld model, are discussed in "Breakable bonds," Section 35.1.9.

Defining surface-based cohesive behavior

You can define surface-based 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.

Surface-based cohesive behavior is modeled within the general contact framework in Abaqus/Explicit and within the contact pair framework in Abaqus/Standard. The details of the surface-based cohesive behavior model are discussed in "Surface-based cohesive behavior," Section 35.1.10.

35.1.2 CONTACT PRESSURE-OVERCLOSURE RELATIONSHIPS

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Mechanical contact properties: overview," Section 35.1.1
- *CONTACT CONTROLS
- *SURFACE BEHAVIOR
- "Creating interaction properties," Section 15.12.2 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Customizing contact controls," Section 15.12.3 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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 slave surface into the master 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 (in Abaqus/Explicit this relationship is available only for the contact pair algorithm);
- 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," Section 35.1.3, for more information. In Abaqus/Standard pressure penetration loads can be applied to model fluid penetrating into the surface between two contacting bodies; see "Pressure penetration loading," Section 35.1.7.

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 35.1.2–1, although the zero-penetration condition may or may 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," Section 36.1.2, and "Contact constraint enforcement methods in Abaqus/Explicit," Section 36.2.3). 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.

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**

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 (in Abaqus/Explicit available only with the contact pair algorithm).

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 slave surfaces on

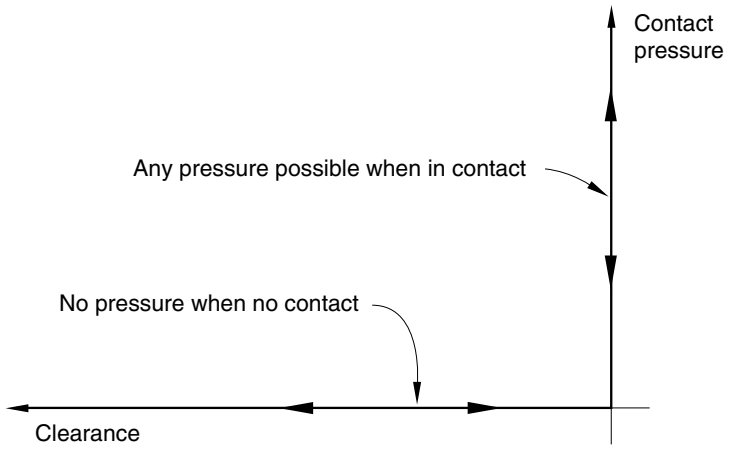


Figure 35.1.2–1 Default pressure-overclosure relationship.

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 slave surfaces on beam-type elements, specify pressure as force per unit area.

When using softened contact relationships that have nonzero pressure at zero overclosure (not allowed with the general contact algorithm) in Abaqus/Explicit, you should be aware that initial, nonequilibrated contact pressures may be present in the analysis (see "Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4).

"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 slave nodes bounce back immediately after impact with the master surface; hence, extensive "chattering" may result, leading to convergence problems and small time increments.

However, softened contact may 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 may 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," Section 36.2.3, 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.

"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 35.1.2–2, 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 the general contact algorithm in Abaqus/Explicit h_1 must be zero. 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 35.1.2–2).

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

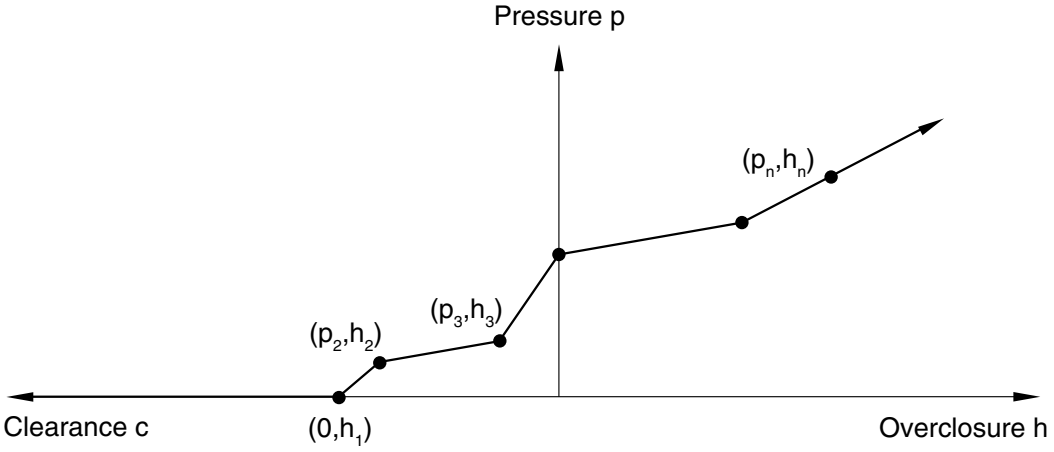


Figure 35.1.2–2 "Softened" pressure-overclosure relationship defined in tabular form.

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 35.1.2–3). The initial stiffness is set equal to the default contact stiffness, k_{dflt}, multiplied by a factor, s_0.

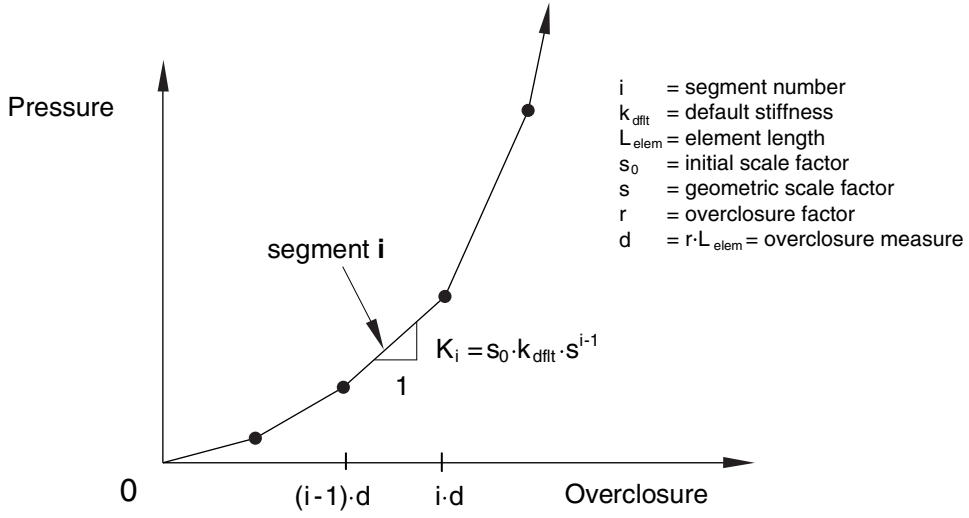


Figure 35.1.2–3 "Softened" scale factor pressure-overclosure relationship.

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 35.1.2–4 illustrates this behavior in Abaqus/Standard. In Abaqus/Explicit this behavior is available only for the contact pair algorithm.

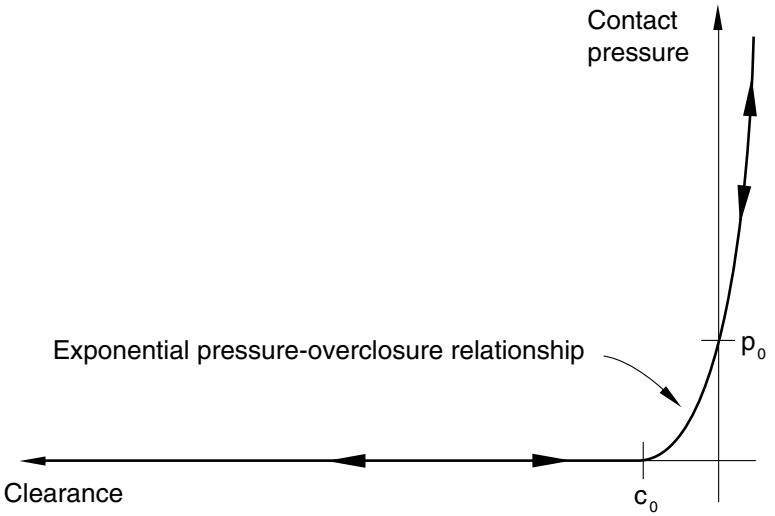


Figure 35.1.2–4 Exponential "softened" pressure-overclosure relationship in Abaqus/Standard.

In Abaqus/Explicit you can specify an optional limit on the contact stiffness that the model can attain, k_{max} (see Figure 35.1.2–5); this limit is useful for penalty contact to mitigate the effect that large stiffnesses have on reducing the stable time increment. 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}

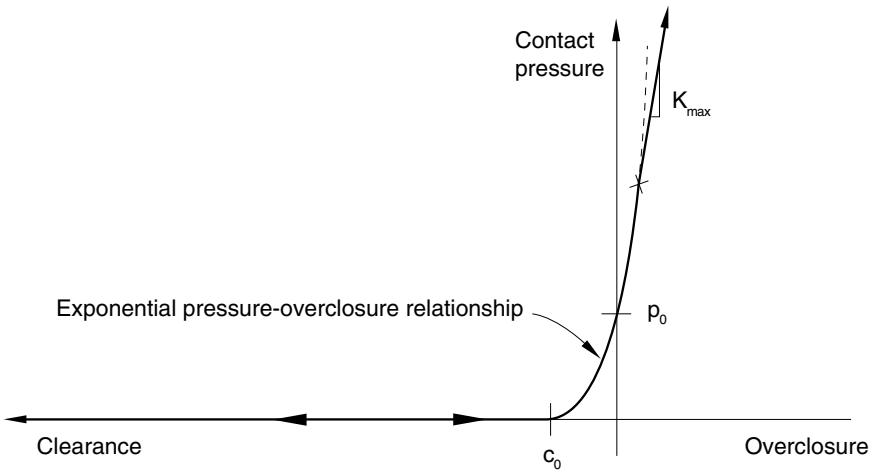


Figure 35.1.2–5 Exponential "softened" pressure-overclosure relationship in Abaqus/Explicit.

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. In Abaqus/Explicit this relationship can be specified only for pure master-slave contact pairs and cannot be used with adaptive meshing or with the general contact algorithm.

The no separation relationship is often used with the rough friction model (see "Frictional behavior," Section 35.1.5) 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.

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**

"Softened" contact with the no separation relationship in Abaqus/Explicit

In Abaqus/Explicit if a softened contact relationship is specified with the no separation relationship, the pressure-overclosure relationship will include tensile behavior. The exponential relationship cannot be used with no separation behavior. For the tabular relationship, a point must be specified on the zero pressure axis, and the slope will continue into the tensile regime following the same slope as the first two data points (see Figure 35.1.2–6). The linear relationship will have a linear tensile pressure-overclosure relationship with the same slope that is used for the compressive behavior.

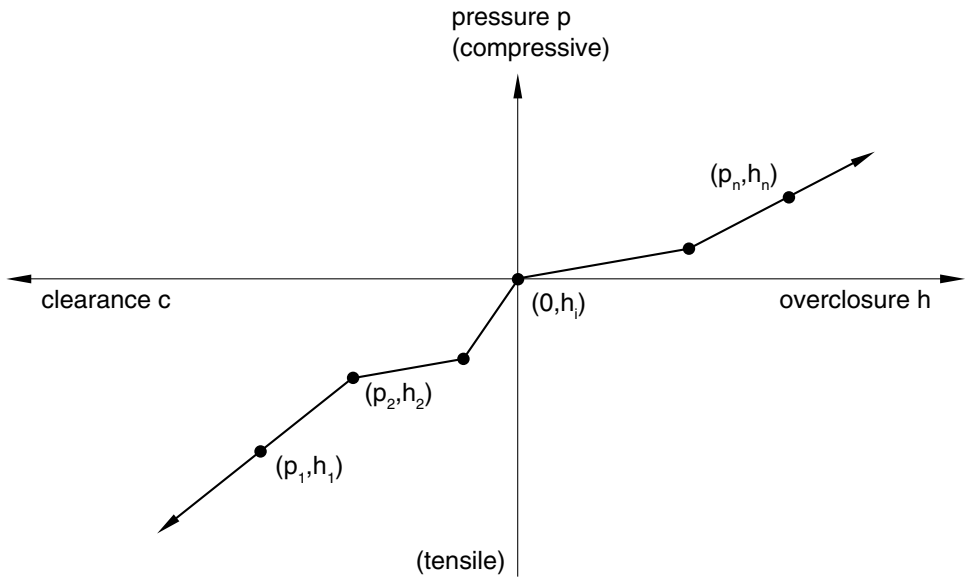


Figure 35.1.2–6 Piecewise linear "softened" pressure-overclosure relationship with tensile behavior in Abaqus/Explicit.

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," Section 4.1.2, and "Output to the output database," Section 4.1.3.

Abaqus/Explicit provides the contact pressure, CPRESS, as output to the output database file (see "Output to the output database," Section 4.1.3, for details).

In the data, results, and output database files the output variable CPRESS gives the viscous damping pressures for an open slave node. This variable also gives the contact pressure for a closed slave node. In printed output a "VD" status indicates that the forces are for viscous damping.

Contours of the contact pressure on the slave surface can be plotted in Abaqus/CAE.

35.1.3 CONTACT DAMPING

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Mechanical contact properties: overview," Section 35.1.1
- *CONTACT DAMPING
- "Creating interaction properties," Section 15.12.2 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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," Section 35.1.2, and the friction models discussed in "Frictional behavior," Section 35.1.5);
- 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," Section 35.1.5, 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" in "Adjusting contact controls in Abaqus/Standard," Section 34.3.6; 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 35.1.3–1. The damping coefficient is defined as a proportionality constant with units of pressure divided by velocity.

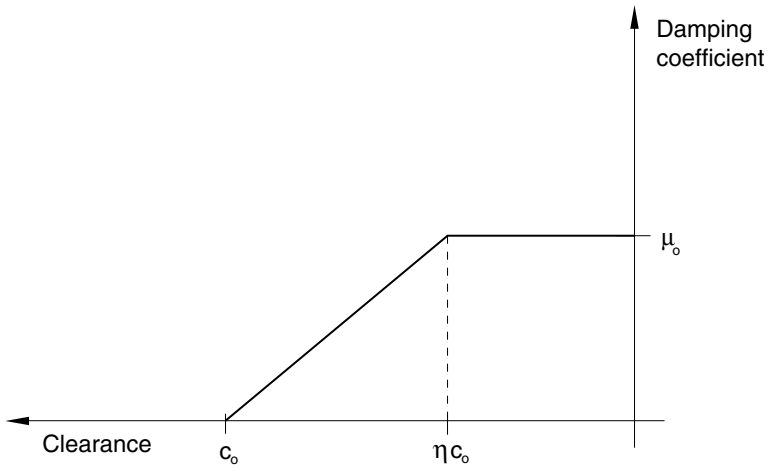


Figure 35.1.3–1 Damping coefficient-clearance relationship for viscous damping in Abaqus/Standard.

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," Section 35.1.2).

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 slave 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
$\mu_0,\ c_0,\ \eta$

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," Section 35.1.5) 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*

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," Section 1.2.1 of the Abaqus Example Problems Manual, 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 35.1.3–2 shows this shape.

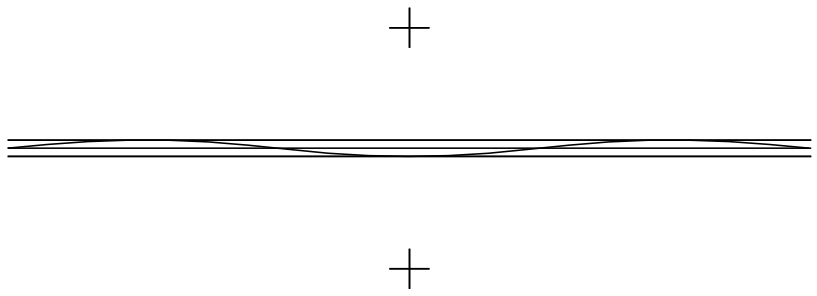


Figure 35.1.3–2 Constrained Euler buckling example for viscous damping.

When the column first buckles, the contact force, F, that the column exerts on one of the rigid surfaces can be approximated as

$$F \approx \frac{\pi^2 h}{4l}(P - P_{crit}),$$

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:

$$\mu_0 = \frac{F}{v} \approx \Delta t \frac{\pi^2}{2l}(P - P_{crit}).$$

This value can be used as a starting value, but different values should be tested.

35.1.4 CONTACT BLOCKAGE

Product: Abaqus/Explicit

References

- "Mechanical contact properties: overview," Section 35.1.1
- "Surface-based fluid cavities: overview," Section 11.5.1
- "Fluid exchange definition," Section 11.5.3
- *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" in "Fluid exchange definition," Section 11.5.3, 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 slave 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," Section 36.2.1).

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," Section 11.5.3), 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.

35.1.5 FRICTIONAL BEHAVIOR

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Mechanical contact properties: overview," Section 35.1.1
- "FRIC," Section 1.1.8 of the Abaqus User Subroutines Reference Manual
- "FRIC_COEF," Section 1.1.9 of the Abaqus User Subroutines Reference Manual
- "VFRIC," Section 1.2.4 of the Abaqus User Subroutines Reference Manual
- "VFRIC_COEF," Section 1.2.5 of the Abaqus User Subroutines Reference Manual
- "VFRICTION," Section 1.2.6 of the Abaqus User Subroutines Reference Manual
- *FRICTION
- *CHANGE FRICTION
- "Creating interaction properties," Section 15.12.2 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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," Section 5.2.3 of the Abaqus Theory Manual), which in Abaqus:
  - in its general form 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, $\bar{\tau}_{max}$, which is the maximum value of shear stress that can be carried by the interface before the surfaces begin to slide;
- include an anisotropic extension of the basic Coulomb friction model in Abaqus/Standard;
- 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 implemented 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," Section 35.1.3, for more information).

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," Section 32.1.2) 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 "Procedures: overview," Section 6.1.1), 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," Section 34.4.3, and "Assigning contact properties for contact pairs in Abaqus/Explicit," Section 34.5.3, 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 slave 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 slip directions for the contact surfaces or contact elements. The slip directions for contact surfaces are defined in "Contact formulations in

Abaqus/Standard," Section 36.1.1, 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," $\bar{\tau}$, for the stick/slip calculations, where $\bar{\tau} = \sqrt{\tau_1^2 + \tau_2^2}$. In addition, Abaqus combines the two slip velocity components into an equivalent slip rate, $\dot{\gamma}_{eq} = \sqrt{\dot{\gamma}_1^2 + \dot{\gamma}_2^2}$. The stick/slip calculations define a surface (see Figure 35.1.5–1 for a two-dimensional representation) in the contact pressure–shear stress space along which a point transitions from sticking to slipping.

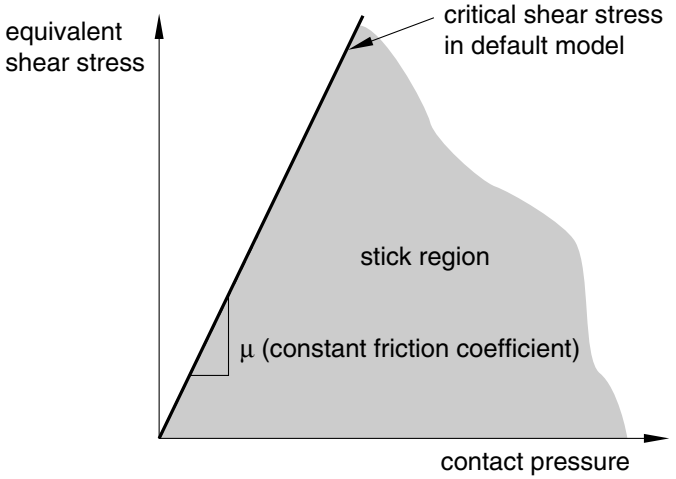


Figure 35.1.5–1 Slip regions for the basic Coulomb friction model.

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.

Using the default model

In the default model you define the coefficient of friction directly as

$$\mu = \mu(\dot{\gamma}_{eq}, p, \bar{\theta}, \bar{f}^{\alpha}),$$

where $\dot{\gamma}_{eq}$ is the equivalent slip rate, $\boldsymbol{p}$ is the contact pressure, $\bar{\theta} = \frac{1}{2}(\theta_A + \theta_B)$ is the average temperature at the contact point, and $\bar{f}^{\alpha} = \frac{1}{2}(f_A^{\alpha} + f_B^{\alpha})$ 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 $\boldsymbol{A}$ and $\boldsymbol{B}$ on the surfaces. Point $\boldsymbol{A}$ is a node on the slave surface, and point $\boldsymbol{B}$ corresponds to the nearest point on the opposing master surface. The temperature and field variables are interpolated along the surface at location $\boldsymbol{B}$. If the master surface consists of a rigid body, the temperature and field variable at the reference node are used. Dependence on $\bar{\theta}$ and $\bar{f}^{\alpha}$ is not available with the general contact algorithm in Abaqus/Explicit.

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.

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.

Input File Usage: *FRICTION, DEPENDENCIES=n
μ, $\dot{\gamma}_{eq}$, p, $\bar{\theta}$, $\bar{f}^{\alpha}$

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 35.1.5–2).

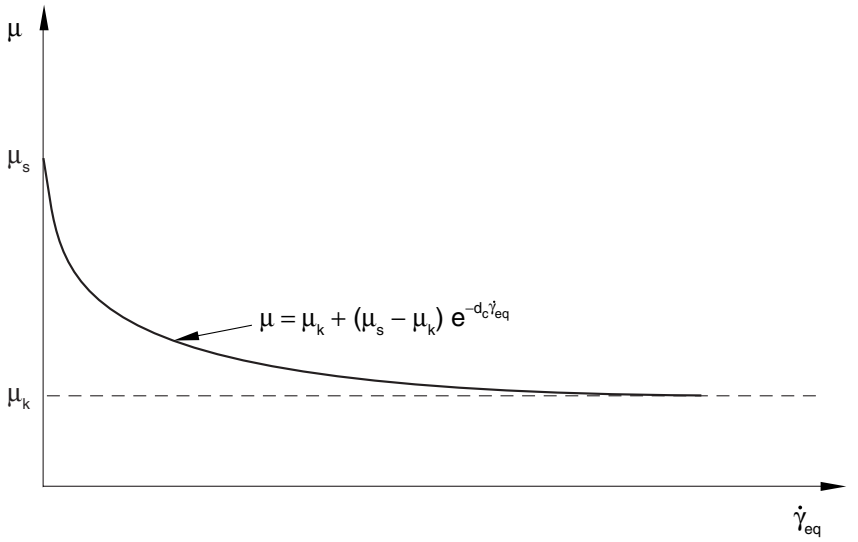


Figure 35.1.5–2 Exponential decay friction model.

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, $(\dot{\gamma}_2, \mu_2)$ (shown in Figure 35.1.5–3), 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, $(\dot{\gamma}_\infty, \mu_\infty)$, to model the assumed asymptotic value of the friction coefficient at infinite velocity. In such a case μ_∞ is chosen such that $(\mu_2 - \mu_\infty)/(\mu_1 - \mu_\infty) = 0.05$.

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**

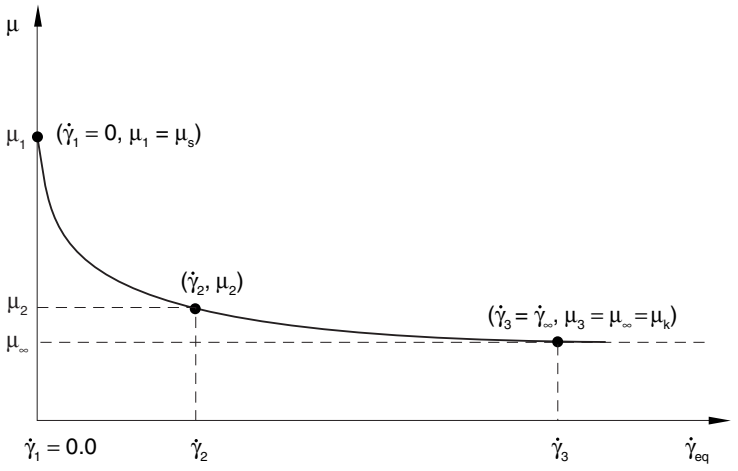


Figure 35.1.5–3 Exponential decay friction model specified with test data points.

Using the optional shear stress limit

You can specify an optional equivalent shear stress limit, $\bar{\tau}_{max}$, so that, regardless of the magnitude of the contact pressure stress, sliding will occur if the magnitude of the equivalent shear stress reaches this value (see Figure 35.1.5–4). A value of zero is not allowed.

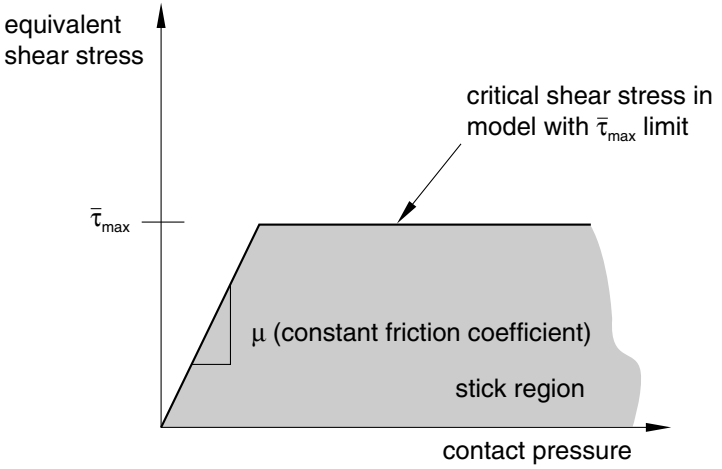


Figure 35.1.5–4 Slip regions for the friction model with a limit on the critical shear stress.

This shear stress limit is typically introduced in cases when the contact pressure stress may 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 $\bar{\tau}_{max}$ 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 $\bar{\tau}_{max}$.

Input File Usage: *FRICTION, TAUMAX=$\bar{\tau}_{max}$

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical→Tangential Behavior**: **Friction formulation: Penalty** or **Lagrange Multiplier**: **Shear Stress**, **Shear stress limit: Specify:** $\bar{\tau}_{max}$

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.

Using the anisotropic friction model in Abaqus/Standard

The anisotropic friction model available in Abaqus/Standard allows for different friction coefficients in the two orthogonal directions on the contact surface. These orthogonal directions coincide with the slip directions defined in "Contact formulations in Abaqus/Standard," Section 36.1.1; and those for contact elements are described in the sections defining contact modeling with those elements. The orientation of the slip directions cannot be changed.

If you indicate that the anisotropic friction model should be used, you must specify two friction coefficients, where μ_1 is the coefficient of friction in the first slip direction and μ_2 is the coefficient of friction in the second slip direction.

The critical shear stress surface (see Figure 35.1.5–5) is an ellipse in τ_1–τ_2 space with the two extreme points being $\tau_1^{crit} = \mu_1 p$ and $\tau_2^{crit} = \mu_2 p$. The size of this 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 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, 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.

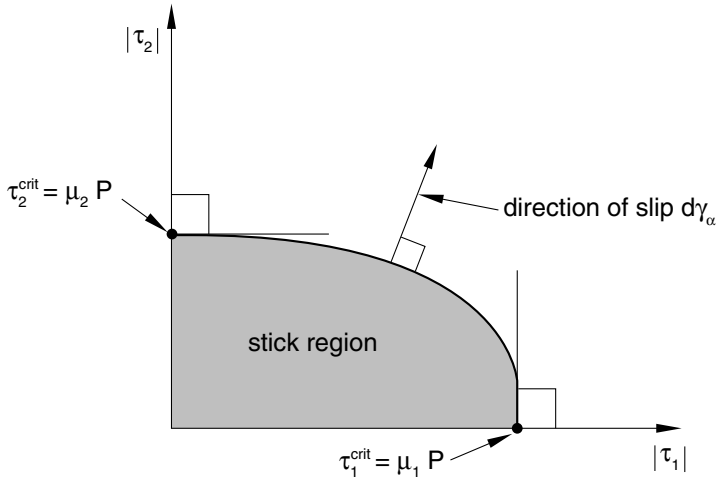


Figure 35.1.5–5 Critical shear stress surface for the anisotropic friction model.

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 may 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" in "Contact pressure-overclosure relationships," Section 35.1.2), 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 may 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 may be finite while in the "sticking" state, as shown in Figure 35.1.5–6.

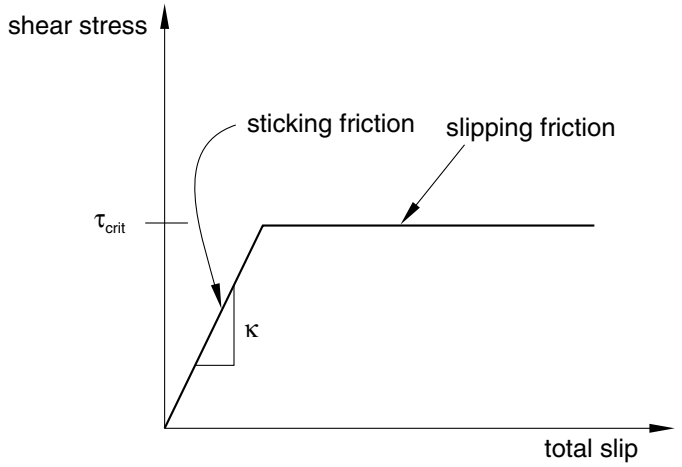


Figure 35.1.5–6 Elastic slip versus shear traction relationship for sticking and slipping friction.

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 may reflect a user-specified physical behavior or may 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," Section 35.1.3. 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 35.1.5–6). 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 (i.e., $\bar{\tau} < \bar{\tau}_{crit}$), the magnitude of sliding is limited to this elastic slip. Abaqus continually adjusts the magnitude of the penalty constraint to enforce this condition.

The stiffness method in Abaqus/Standard requires the selection of an allowable elastic slip, γ_i. Using a large γ_i 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_i. If γ_i is chosen very small, convergence problems may occur; in that case, it may 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_i as a small fraction of the "characteristic contact surface length," $\bar{l}_i$, and scans all of the facets of all the slave surfaces when calculating $\bar{l}_i$. Abaqus/Standard reports the value of $\bar{l}_i$ 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" in "Output," Section 4.1.1). The allowable elastic slip is given as $\gamma_i = F_f \bar{l}_i$, where F_f is the slip tolerance; the default value of F_f is 0.005.

This method of calculating the allowable elastic slip is used for all analysis procedures in Abaqus/Standard except steady-state transport analysis ("Steady-state transport analysis," Section 6.4.1), in which the penalty constraint is based on a maximum allowable slip rate, $\dot{\gamma}_i$. The maximum slip rate is calculated as

$$\dot{\gamma}_i = F_f 2\omega R,$$

where ω is the angular spinning rate and R is the radius of the rolling structure.

Cases in which the default elastic slip value may not be suitable

In certain situations the default value for the allowable elastic slip may not be suitable. For example, slave 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 $\bar{l}_i$. 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 $\bar{l}_i$ using all of 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 $\bar{l}_i$. 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 $\bar{l}_i$ may 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 methods for modifying the allowable elastic slip. One method is to specify γ_i directly; the other is to specify the slip tolerance, F_f. Some analyses call for nondefault γ_i 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 provide the absolute magnitude of γ_i directly. Specify a reasonable value for the relative displacement that may 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, $\dot{\gamma}_i$.

The specified allowable elastic slip will be 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_i
ASURF, BSURF	DEFAULT	$F_f\bar{l}_i$
CSURF, BSURF	NONDEF	0.1

In the **DEFAULT** contact property definition no value for γ_i is specified, so the allowable elastic slip used for the friction interaction between **ASURF** and **BSURF** would be the default value $F_f\bar{l}_i$. In the **NONDEF** contact property definition a value of 0.1 is specified for γ_i, which will be the allowable elastic slip used for the friction interaction between **CSURF** and **BSURF**.

Input File Usage: *FRICTION, ELASTIC SLIP=γ_i

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical→Tangential Behavior**: **Friction formulation: Penalty** or **Static-Kinetic Exponential Decay**: **Elastic Slip**, **Absolute distance:** γ_i

Changing the default slip tolerance

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: *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**, **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 (i.e., $\bar{\tau} < \bar{\tau}_{crit}$), 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," Section 36.2.3).

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," Section 34.4.5, and "Contact controls for contact pairs in Abaqus/Explicit," Section 34.5.5.

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 $\bar{\tau} = \bar{\tau}_{crit}$. 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 may 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.

Input File Usage: *FRICTION, LAGRANGE

Abaqus/CAE Usage: 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," Section 36.2.3). 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 slave node and either the nodes of the master 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 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," Section 35.1.6, 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 isotropic friction

Abaqus provides a simple way to specify complex isotropic frictional behavior when the expression for the friction coefficient can be defined explicitly. You need only to specify the friction coefficient, and Abaqus will compute the resulting frictional forces. Abaqus/Standard provides user subroutine **FRIC_COEF** and Abaqus/Explicit provides user subroutine **VFRIC_COEF** for this purpose. **VFRIC_COEF** can be used only with general contact.

Input File Usage: *FRICTION, USER=COEFFICIENT, PROPERTIES=*p*

Abaqus/CAE Usage: User subroutines **FRIC_COEF** and **VFRIC_COEF** are not supported in Abaqus/CAE.

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" in "Procedures: overview," Section 6.1.1.

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," Section 35.2.1.

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 slave surface that use a surface interaction model containing frictional properties. The shear stresses, CSHEAR1 and CSHEAR2, are given in the two orthogonal slip directions, which are constructed on the master surface (see "Contact formulations in Abaqus/Standard," Section 36.1.1). There is only one slip direction in two-dimensional problems. Details about how to request contact surface variable output are given in "Defining contact pairs in Abaqus/Standard," Section 34.3.1, and "Defining contact pairs in Abaqus/Explicit," Section 34.5.1.

Contour plots of these variables can also be plotted in Abaqus/CAE.

Additional reference

- 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.

35.1.6 USER-DEFINED INTERFACIAL CONSTITUTIVE BEHAVIOR

Products: Abaqus/Standard Abaqus/Explicit

References

- "UINTER," Section 1.1.38 of the Abaqus User Subroutines Reference Manual
- "VUINTER," Section 1.2.15 of the Abaqus User Subroutines Reference Manual
- "VUINTERACTION," Section 1.2.16 of the Abaqus User Subroutines Reference Manual
- *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 slave node (or points on the slave 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 slave node with respect to the master 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 slave 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 slave surface with respect to the master 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 slave 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 slave surface with respect to the corresponding closest point on the master surface, as well as the incremental relative motion between these two points. Values of temperature and field variables at the constraint point on the slave surface and the corresponding closest point on the master 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 slave 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 slave node (or a point on the slave 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 VUINTERACTION is used

A surface interaction is considered active if the interacting surfaces are within a separation distance called the tracking thickness. Abaqus/Explicit uses an internal default value for the tracking thickness. Alternatively, you can specify the tracking thickness in conjunction with a user-defined surface interaction model. In this case contacting surfaces whose proximity is 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 slave surface at each iteration of every increment. User subroutine **VUINTER** is called in every time increment for each master-slave 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 slave 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, $\partial \triangle \boldsymbol{\sigma} / \partial \triangle \mathbf{u}$, is not symmetric, you should invoke the unsymmetric equation solution capability in Abaqus/Standard (see "Procedures: overview," Section 6.1.1).

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" in "Output," Section 4.1.1), 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 slave 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" in "Procedures: overview," Section 6.1.1. 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" in "Contact constraint enforcement methods in Abaqus/Explicit," Section 36.2.3.

When **VUINTER** is used and balanced master-slave contact is specified (i.e., 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 slave surface. The forces and fluxes defined in **VUINTER** will be multiplied by the weight value for the master-slave 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 master-slave 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," Section 1.2.15 of the Abaqus User Subroutines Reference Manual, and "VUINTERACTION," Section 1.2.16 of the Abaqus User Subroutines Reference Manual, 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 slave nodes are in contact. In the case of **VUINTER**, if only a fraction of the slave 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 slave 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 slave 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** can be used in conjunction with **VUINTER** to define the mechanical constitutive behavior 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.

35.1.7 PRESSURE PENETRATION LOADING

Products: Abaqus/Standard Abaqus/CAE

References

- *PRESSURE PENETRATION
- *SURFACE
- *CONTACT PAIR
- "Defining pressure penetration," Section 15.13.13 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Pressure penetration loads simulated with contact pairs:

- model the penetration of fluid between two contacting structures; and
- allow the fluid to penetrate from multiple locations on the surface.

Defining pressure penetration loads between contacting bodies

Distributed pressure penetration loads allow for the simulation of fluid penetrating into the surface between two contacting bodies and application of the fluid pressure normal to the surfaces. Element-based contact surfaces are used to model the interactions between the bodies (see "Contact interaction analysis: overview," Section 34.1.1). The surfaces are modeled as slave and master contact surfaces (see "Contact formulations in Abaqus/Standard," Section 36.1.1).

Any contact formulation can be used.

The bodies forming the joint may both be deformable, as would be the case with threaded connectors; or one may be rigid, as would occur when a soft gasket is used as a seal between stiffer structures. You specify the nodes exposed to the fluid pressure, the magnitude of the fluid pressure, and the critical contact pressure below which fluid penetration starts to occur. See "Pressure penetration loading with surface-based contact," Section 6.4.1 of the Abaqus Theory Manual, for more details.

Input File Usage: *PRESSURE PENETRATION, SLAVE=*slave1*, MASTER=*master1*
slave surface node or node set, *master surface node or node set*, *magnitude*, *critical contact pressure*

If a node set is specified, it can contain only one node in two dimensions; in three dimensions it can contain any number of nodes.

Abaqus/CAE Usage: Interaction module:
Create Interaction: **Surface-to-surface contact (Standard)**, **Name:** *contact_interaction_name*; select master and slave surfaces
Create Interaction: **Pressure penetration**; **Contact interaction:** *contact_interaction_name*, **Region on Master:** select face, edge, or point,

Region on Slave: select face, edge, or point, **Critical Contact Pressure:** *critical contact pressure*, **Fluid Pressure:** *magnitude*

Specifying a pressure penetration criterion

A single slave-node-based penetration criterion is used. Fluid will penetrate into the surface between the contacting bodies from one or multiple locations, which are exposed to the fluid, until a point is reached where the contact pressure is greater than the specified critical value, cutting off further penetration of the fluid.

Specifying a penetration time for the fluid pressure

When the fluid pressure penetration criterion is satisfied, the fluid pressure is applied normal to the surfaces. If the full current fluid pressure is applied immediately, the resulting large changes in the strains near the contact surfaces can cause convergence difficulties. For large-strain problems severe mesh distortion can also occur. To ensure a smooth solution, the fluid pressure is ramped up linearly over a time period from zero pressure penetration load to the full current magnitude.

You can specify the time period taken for the fluid pressure penetration load to reach the full current magnitude on newly penetrated surface segments. If the accumulated increment size, measured immediately after the penetration, is greater than the penetration time, the full current fluid pressure penetration load will be applied; otherwise, the fluid pressure on the newly penetrated surface segments is ramped up linearly to the current magnitude over the penetration time period, possibly over a number of increments. When the penetration time is equal to 0, the current fluid pressure is applied immediately once the fluid pressure penetration criterion is satisfied. The default penetration time is chosen to be 0.001 of the total step time. The penetration time is ignored in a linear perturbation analysis.

Input File Usage: ∗PRESSURE PENETRATION, PENETRATION TIME=*n*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Pressure penetration**; **Penetration time:** *n*

Specifying the nodes exposed to the fluid pressure

The fluid can penetrate from either one or multiple locations of the surface. You must identify a node or node set on the slave surface of the contacting bodies that defines where the surface is exposed to the fluid pressure. In two dimensions if the master surface is not an analytical rigid surface (see "Analytical rigid surface definition," Section 2.3.4), you must also identify a node or node set on the master surface that defines where the surface is exposed to the fluid pressure. You can specify multiple nodes or node sets if multiple locations of the surface are exposed to the fluid. These nodes or node sets are always subjected to the pressure penetration load if they are on the slave surface, regardless of their contact status. The fluid then starts to penetrate into the surface between the two contacting bodies from these nodes or node sets.

Specifying the applied fluid pressure

You must define the reference magnitude of the fluid pressure. You can define the variation of the fluid pressure during a step by referring to an amplitude curve. By default, the reference magnitude is applied

immediately at the beginning of the step or ramped up linearly over the step, depending on the amplitude variation assigned to the step (see "Procedures: overview," Section 6.1.1).

The fluid pressure penetration load will be applied to the element surface based on the pressure penetration criterion at the beginning of an increment and will remain constant over that increment even if the fluid penetrates further during that increment. A nodal integration scheme is used to integrate the distributed fluid pressure penetration load over an element in two dimensions, while in three dimensions Gauss integration scheme is used; the variation of the distributed fluid pressure over an element will be determined by the load magnitudes at the element's nodes.

Input File Usage: Use the following option to define the variation of the fluid pressure during a step:

*PRESSURE PENETRATION, AMPLITUDE=*name*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Pressure penetration**; **Amplitude:** *name*

Removing or modifying the pressure penetration loads

After pressure penetration loads are applied to the element surfaces, they will not be removed automatically even when contact between the surfaces is reestablished. At each new step the fluid pressure penetration loading, however, can be modified or completely redefined in a manner similar to the way that distributed loads can be defined (see "Applying loads: overview," Section 32.4.1).

Input File Usage: Use the following option to modify the fluid pressure penetration loads that were applied in previous steps:

*PRESSURE PENETRATION, OP=MOD (default)

In this case the slave nodes exposed to the fluid pressure must be specified on the data lines. If the master surface is not an analytical rigid surface, the master nodes exposed to the fluid pressure must also be specified on the data lines for planar or axisymmetric models.

Use the following option to remove all fluid pressure penetration loads and, optionally, to specify new fluid pressure penetration loads:

*PRESSURE PENETRATION, OP=NEW

When OP=NEW is used to remove all fluid pressure penetration loads, no data line is needed. However, when OP=NEW is used to specify new fluid pressure penetration loads, the nodes exposed to the fluid pressure must be specified on the data lines. OP=NEW must be used when defining new exposed nodes. In addition, when OP=NEW is used to re-specify a previously defined pressure penetration load, the fluid pressure loading will revert to its last known configuration first, even if the contact status has subsequently changed.

Abaqus/CAE Usage: Use the following option to modify a fluid pressure penetration that was applied in a previous step:

Interaction module: **Interaction Manager**: select interaction, **Edit**

Use the following option to remove a fluid pressure penetration that was applied in a previous step:

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

Specifying a critical mechanical contact pressure

To account for the asperities on the contacting surfaces, a critical contact pressure, below which fluid penetration starts to occur, is introduced. The higher this value, the easier the fluid penetrates. The default value of the critical contact pressure is zero, in which case fluid penetration occurs only if contact is lost.

Use in linear perturbation analysis

Linear perturbation analyses can be performed from time to time during a fully nonlinear analysis by including linear perturbation steps between the general analysis steps. Because contact conditions cannot change during a linear perturbation analysis, the fluid will not penetrate further into the surface and it remains as it was defined in the base state. The fluid pressure magnitude applied in the previous general analysis step, however, can be modified during a linear perturbation analysis step. In matrix generation (see "Generating matrices," Section 10.3.1) and steady-state dynamic analyses (direct or modal—see "Direct-solution steady-state dynamic analysis," Section 6.3.4, and "Mode-based steady-state dynamic analysis," Section 6.3.8) you can specify both the real (in-phase) and imaginary (out-of-phase) parts of the loading.

Input File Usage: Use the following option to define the real (in-phase) part of the loading:

*PRESSURE PENETRATION, REAL (default)

Use the following option to define the imaginary (out-of-phase) part of the loading:

*PRESSURE PENETRATION, IMAGINARY

The REAL or IMAGINARY parameters are ignored in all procedures other than steady-state dynamics.

Abaqus/CAE Usage: Use the following option to define the real (in-phase) part of the loading:

Interaction module: **Create Interaction**: **Pressure penetration**; **Fluid Pressure (Real)**

Use the following option to define the imaginary (out-of-phase) part of the loading:

Interaction module: **Create Interaction**: **Pressure penetration**; **Fluid Pressure (Imaginary)**

Limitations with pressure penetration loads

Each slave surface subjected to pressure penetration loading must be continuous and cannot be a closed loop. Pressure penetration loading cannot be used with a node-based slave surface. The pressure penetration load applied at any increment is based on the contact status at the beginning of that

increment. You should, therefore, be careful in interpreting the results at the end of an increment during which the contact status has changed. Small time increments are recommended to obtain accurate results.

When pressure penetrates into contacting bodies between an analytical rigid surface and a deformable surface, no pressure penetration load will be applied to the analytical rigid surface. The reference node on the analytical rigid surface should, therefore, be constrained in all directions. To account for the effect of fluid pressure penetration loads on the rigid surface, the analytical rigid surface should be replaced with an element-based rigid surface.

When fluid with different pressure loads penetrates into an element simultaneously from multiple locations on a surface, the maximum value of the fluid pressure loads is applied to the element.

In large-displacement analyses pressure penetration loads introduce unsymmetric load stiffness matrix terms. Using the unsymmetric matrix storage and solution scheme for the analysis step may improve the convergence rate of the equilibrium iterations. See "Procedures: overview," Section 6.1.1, for more information on the unsymmetric matrix storage and solution scheme.

Output

You can request the fluid pressure load, PPRESS, at the nodes on the slave surface as surface output to the data, results, and output database files (see "Surface output from Abaqus/Standard" in "Output to the data and results files," Section 4.1.2, and "Surface output in Abaqus/Standard and Abaqus/Explicit" in "Output to the output database," Section 4.1.3).

35.1.8 INTERACTION OF DEBONDED SURFACES

Product: Abaqus/Standard

References

- "Contact pressure-overclosure relationships," Section 35.1.2
- "Frictional behavior," Section 35.1.5
- "Thermal contact properties," Section 35.2.1
- "Pore fluid contact properties," Section 35.4.1
- *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," Section 11.4.3.

When two initially bonded surfaces start to debond:

- the debonded slave surface nodes are released and can move freely;
- the tractions acting on the slave 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 slave 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 slave 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.

35.1.9 BREAKABLE BONDS

Product: Abaqus/Explicit

References

- "Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2
- *BOND
- *SURFACE INTERACTION
- *CONTACT PAIR

Overview

Breakable bonds, such as spot welds, between surfaces:

- can be defined only at the nodes of the slave surface of a pure master-slave contact pair;
- can be defined only in the first step of a simulation;
- constrain the slave node to the master surface until the failure criterion of the bond is met;
- are designed to provide a simple simulation of spot weld failure under relatively monotonic straining, such as occurs during an impact of a vehicle structure;
- do not constrain the rotational degrees of freedom at the node;
- use either a time to failure or a damaged failure model to simulate the postfailure response of the bonds;
- use the default contact property model ("Mechanical contact properties: overview," Section 35.1.1) once the bonds have been broken; and
- can be used only between two deformable surfaces with the kinematic contact pair algorithm.

Specifying spot welds for a contact pair

A contact pair that contains spot welds must be a pure master-slave contact pair; therefore, spot welds cannot be used with single-surface contact. If the contact pair consists of two deformable surfaces, Abaqus/Explicit would normally use a balanced master-slave contact pair. In such situations you must specify a weighting factor (see "Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2) to define a pure master-slave contact pair. Contact pairs containing spot welds must be defined in the first step of a simulation. The spot welds are located at the nodes of the slave surface of the contact pair.

Spot welds can also be modeled more accurately using fasteners instead of breakable bonds. Fasteners have the advantage of being mesh independent in their definition and are convenient for defining point-to-point connections between two or more surfaces with the capability to model plasticity, damage, and failure behavior. However, fasteners are intended to be used in three dimensions; therefore, the fastener method cannot be used to specify spot welds for contact pairs in a two-dimensional case.

If non-breakable bonds (rigid spot welds) are to be modeled, it is recommended that you use the mesh-independent spot weld feature ("Mesh-independent fasteners," Section 33.3.4).

All of the slave nodes which are bonded to a master surface can be grouped together into a node set.

Input File Usage: Use all of the following options:

∗CONTACT PAIR, MECHANICAL CONSTRAINT=KINEMATIC, INTERACTION=*interaction_property_name*
∗SURFACE INTERACTION, NAME=*interaction_property_name*
∗BOND
node_set_name, …

Adjustments to the initial positions of the bonded nodes

Nodes that are bonded to a master surface with spot welds should be defined so that they contact the surface in the model's initial configuration. If the bonded nodes are not in contact initially, Abaqus/Explicit will enforce the bonded constraint by prescribing strain-free displacements to those nodes. The nodes will begin the simulation exactly in contact with the master surface. If the spot welds are defined incorrectly, this automatic adjustment of the nodes may cause the analysis to end immediately as a result of excessive initial distortion of elements that are connected to the bonded nodes.

Forces carried by a spot weld

Abaqus assumes that a spot weld carries a force normal to the surface onto which the node is welded, F^n, and two orthogonal shear forces tangent to the surface, F^s_α, $\alpha = 1, 2$. The magnitude of the resultant shear force, F^s, is defined as $\sqrt{(F^s_1)^2 + (F^s_2)^2}$. The normal force is positive in tension.

A spot weld is assumed to be so small that it carries no moments or torque. As a result, spot welds do not impose any constraints on rotational degrees of freedom.

Defining the failure criterion for the spot welds

The failure criterion for a spot weld is defined as

$$\left(\frac{\max(F^n, 0)}{F^n_f}\right)^2 + \left(\frac{F^s}{F^s_f}\right)^2 \leq 1.0,$$

where

F^n_f	is the force required to cause failure in tension (Mode I loading),
F^s_f	is the force required to cause failure in pure shear (Mode II loading), and
F^n and F^s	are defined above.

A typical yield surface for spot welds is shown in Figure 35.1.9–1. By specifying a very large value for either F^n_f or F^s_f, the yield criteria of the spot welds can be made independent of either shear forces or normal forces, as shown in Figure 35.1.9–2.

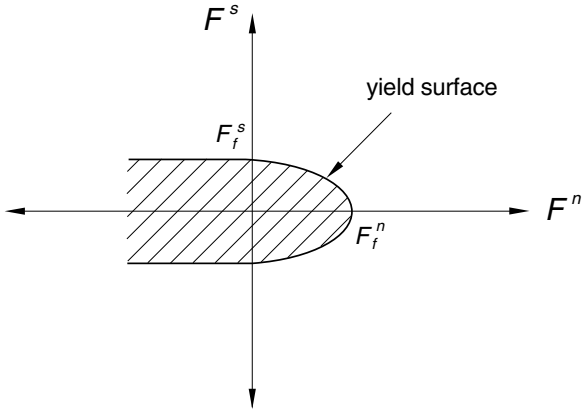


Figure 35.1.9–1 Typical yield surface for spot welds.

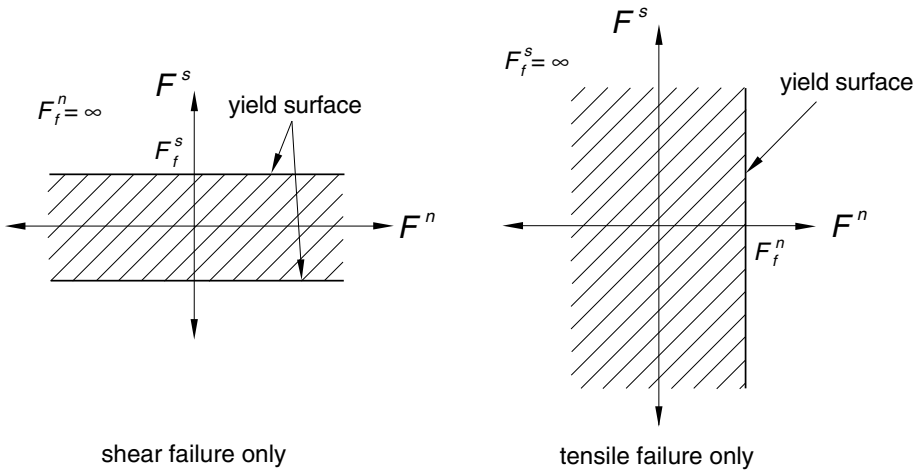


Figure 35.1.9–2 Degenerate yield surfaces for spot welds.

Input File Usage: *BOND
node_set_name, F^n, F^s

Spot weld forces sometimes exhibit significant noise, which can cause the spot weld to reach its failure criterion when a filtered solution of the spot weld forces would still be well within the strength limits of the spot weld. This is characterized by a noisy time history of the BONDSTAT variable and can correspond to an unrealistically early onset of failure of a spot weld. Two models for deterioration of a spot weld after the onset of failure are discussed below: a time to failure model and a postfailure damage model. With the time to failure model a single, spurious spike in the constraint force history that just

exceeds the spot weld strength will lead to complete failure of the spot weld. The postfailure damage model may mitigate the effects of noise in the spot weld force.

Defining the postfailure behavior of the spot welds

Once the constraint forces on a spot weld exceed the failure criterion, the spot weld fails and deteriorates until the weld is broken completely. The behavior of the spot weld during this deterioration process can be simulated using either a damaged failure model or by linearly reducing the constraint forces to zero over a specified time period. With either model, the applied constraint forces from a spot weld are limited by the size of the yield surface as defined by the failure criterion. Deterioration of the spot weld is modeled by shrinking the yield surface to zero while retaining its original shape.

If the predicted constraint forces exceed the yield surface, the applied forces are calculated using a radial flow rule to return to the yield surface.

After complete failure, the node behaves like the rest of the slave nodes in the contact pair. The node may recontact the master surface, but the weld plays no further role.

Defining the time to failure model

You specify the time to failure, T_f, which is the time required for the spot weld to fail completely after the initial failure criterion has been exceeded. Once failure is detected, the weld constraint is relaxed linearly over the time T_f. Abaqus/Explicit shrinks the yield surface to zero over the time period T_f:

$$\left(\frac{\max(F^n,0)}{F_f^n}\right)^2+\left(\frac{F^s}{F_f^s}\right)^2\le\left(1.0-\frac{t}{T_f}\right)^2,$$

where t is the time since Abaqus/Explicit detected initial failure of the weld.

Input File Usage: ∗BOND
node_set_name, F^n, F^s, d_b, T_f

Defining the postfailure damage model

As stated above, if the predicted constraint forces exceed the failure criterion, the forces carried by the spot weld are calculated using a radial flow rule to return to the yield surface. Since the forces in the weld in this case are less than the constraint forces required to constrain the welded node on the master surface, the welded node will move relative to the master surface. The work expended during this relative motion is used to determine how the yield surface degrades.

During failure the behavior of the weld is assumed to be such that any stretching of the weld in the normal direction, or any shearing of the weld, dissipates energy. Abaqus/Explicit assumes a linear force-displacement relationship after failure, thus resulting in the behaviors sketched in Figure 35.1.9–3 when the weld is subjected to pure Mode I or pure Mode II loading. More general loadings create combinations of these responses.

You define the amount of energy that the weld can dissipate in Mode I and Mode II by specifying the breakage displacements in the normal and shear directions under pure Mode I and Mode II loading, u_f^n and u_f^s.

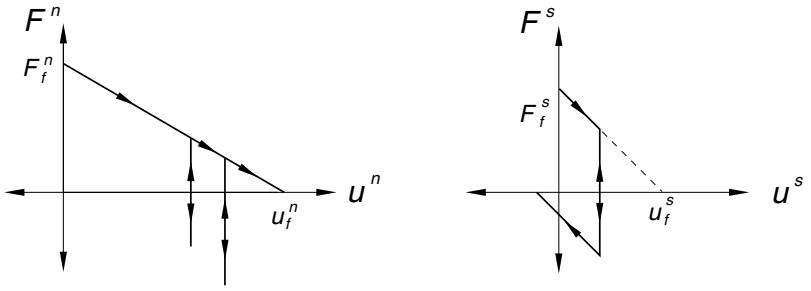


Figure 35.1.9–3 Typical postfailure behavior in pure tension/compression (Mode I) and in pure shear (Mode II).

Using these linear force-displacement relationships, the failure criterion for the damaged failure model is

$$\left(\frac{\max(F^n,0)}{F_f^n}\right)^2+\left(\frac{F^s}{F_f^s}\right)^2\le 1.0-\frac{E^I}{E_f^I}-\frac{E^{II}}{E_f^{II}},$$

where

E^I is the energy expended in Mode I;
E^{II} is the energy expended in Mode II;
E_f^I is the breakage energy in Mode I, which is calculated as $F_f^n \cdot u_f^n/2$; and
E_f^{II} is the breakage energy in Mode II, which is calculated as $F_f^s \cdot u_f^s/2$.

Input File Usage: *BOND
node_set_name, F^n, F^s, d_b, , u_f^n, u_f^s

Post-yield surface interactions in spot welds

Any friction, contact damping, or softening defined at the spot weld will not affect the analysis until the weld is broken completely; i.e., until the failure surface has shrunk to zero.

Bead size of the spot weld

The initial bead size of the spot weld, d_b, is taken into account by offsetting the slave surface node associated with the spot weld from the master surface by an amount equal to the bead size during the penetration calculations. A master or slave surface defined on shell or membrane elements is itself offset from the midplane of the element by the half-thickness of the shell or membrane.

If the damaged failure model is chosen to characterize the postfailure behavior, the size of the spot weld bead may grow due to tensile yielding of the spot weld. The size of the spot weld is equal to the sum of d_b and the U^n accumulated after the failure of the spot weld. After the weld has broken, the

size of the bead at breakage is taken into account for subsequent contact between the weld node and the master surface.

Available output for spot welds

You can examine the forces carried by spot welds in Abaqus/CAE by generating a vector plot of the reaction forces on the surface (output variable CFORCE). Two output variables specifically related to spot welds, the bond status and bond load, are available for use in Abaqus/CAE. These variables can be written as history output to the output database (**.odb**) file. They can be used in *X*–*Y* plots in Abaqus/CAE.

Definition of bond status

The bond status (output variable BONDSTAT) is a measure of how close a spot weld is to complete failure. The bond status varies between 0.0 and 1.0 and is defined to be

$$1.0 - \frac{t}{T_f}$$

if the time to failure postfailure model is chosen or

$$\sqrt{1.0 - \frac{E^I}{E_f^I} - \frac{E^{II}}{E_f^{II}}}$$

if the damaged failure model is chosen. With either model, the bond status is equal to 1.0 before the spot weld fails.

Definition of bond load

The bond load (output variable BONDLOAD) is a measure of how close the current constraint forces at a spot weld are to its failure surface. The value of the bond load also varies between 0.0 and 1.0 and is defined to be

$$\sqrt{\frac{\left(\frac{\max(F^n,0)}{F_f^n}\right)^2 + \left(\frac{F^s}{F_f^s}\right)^2}{\left(1.0 - \frac{E^I}{E_f^I} - \frac{E^{II}}{E_f^{II}}\right)}}$$

if the damaged failure model is chosen. For the time to failure model, the bond load is defined to be

$$\sqrt{\left(\frac{\max(F^n,0)}{F_f^n}\right)^2 + \left(\frac{F^s}{F_f^s}\right)^2}$$

prior to failure. Then, the bond load is 1.0 from the moment of first yield until total failure, at which point the bond load becomes 0.0.

Example: Spot welds and output requests

The spot-welded nodes in node set **WELDS** are a subset of the nodes on surface A, which is the slave surface of the pure master-slave contact pair.

```
*NSET, NSET=WELDS
node set definition
*CONTACT PAIR, MECHANICAL CONSTRAINT=KINEMATIC,
INTERACTION=A TO B, WEIGHT=0.
slave surface A, master surface B
*SURFACE INTERACTION, NAME=A TO B
*BOND
WELDS, F_f^n, F_f^s, d_b, T_f, u_f^n, u_f^s
*OUTPUT, HISTORY, TIME INTERVAL=0.001
*CONTACT OUTPUT, NSET=WELDS
BONDSTAT, BONDLOAD
```

Here T_f must be specified if the time to failure model is used, or u_f^n and u_f^s must be specified if the damaged failure model is chosen.

35.1.10 SURFACE-BASED COHESIVE BEHAVIOR

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Progressive damage and failure," Section 23.1.1
- "Defining the constitutive response of cohesive elements using a traction-separation description," Section 31.5.6
- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- "Defining general contact interactions in Abaqus/Explicit," Section 34.4.1
- "Mechanical contact properties: overview," Section 35.1.1
- "Crack propagation analysis," Section 11.4.3
- *COHESIVE BEHAVIOR
- *SURFACE INTERACTION
- *DAMAGE INITIATION
- *DAMAGE EVOLUTION
- *DAMAGE STABILIZATION
- *FRACTURE CRITERION
- "Specifying cohesive behavior properties for mechanical contact property options" in "Defining a contact interaction property," Section 15.14.1 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Specifying cohesive damage properties for mechanical contact property options" in "Defining a contact interaction property," Section 15.14.1 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

The features described in this section allow the specification of generalized traction-separation behavior for surfaces. This behavior offers capabilities that are very similar to cohesive elements that are defined using a traction-separation law (see "Defining the constitutive response of cohesive elements using a traction-separation description," Section 31.5.6). However, surface-based cohesive behavior is typically easier to define and allows simulation of a wider range of cohesive interactions, such as two "sticky" surfaces coming into contact during an analysis.

Surface-based 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 may be more appropriate to model the response using conventional cohesive elements (see "Defining the constitutive response of cohesive elements using a continuum approach," Section 31.5.5).

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).

Surface-based 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 (i.e., surfaces or parts of surfaces that are not initially in contact may bond on coming into contact; subsequently the bond may damage and fail);
- can be restricted to surface regions that are initially in contact and, in Abaqus/Standard, to portions of 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 crierion);
- allows specification of post-failure cohesive behavior if failed nodes re-enter contact;
- is implemented within the general contact algorithmic framework in Abaqus/Explicit and within the contact pair framework in Abaqus/Standard;
- can be used to enforce "rough friction" surface interactions, the "no separation" contact relationship, or a combined "no separation and rough friction" behavior within the general contact framework in Abaqus/Explicit;
- is enforced only for node-to-face contact interactions in Abaqus/Explicit and is not available for edge-to-edge and node-to-analytical rigid surface contact interactions;
- cannot be used in a coupled Eulerian-Lagrangian analysis in Abaqus/Explicit; and
- can be used for all Abaqus/Standard contact formulations except the finite sliding, surface-to-surface formulation.

Defining cohesive behavior in Abaqus/Explicit

Cohesive behavior in Abaqus/Explicit is defined as part of the surface interaction properties that are assigned to the applicable surfaces. General contact must be defined for the model.

Input File Usage: Use the following options to define cohesive behavior between two surfaces in a general contact definition:

```
*SURFACE INTERACTION, NAME=name
*COHESIVE BEHAVIOR
*CONTACT
*CONTACT PROPERTY ASSIGNMENT
surface1, surface2, name
```

Abaqus/CAE Usage: Use the following option to define cohesive behavior between two surfaces:

Interaction module: contact property editor: **Mechanical→Cohesive Behavior**

Use the following option to define contact between two surfaces:

Interaction module: interaction editor: **General contact (Explicit)**: specify **Contact interaction property**

Contact formulation for cohesive behavior in Abaqus/Explicit

In Abaqus/Explicit overconstraints can arise in certain situations if the balanced master-slave formulation is enforced in addition to the cohesive constraint. To prevent this from occurring, a pure master-slave formulation is enforced for surfaces with cohesive behavior in Abaqus/Explicit. If cohesive behavior is defined between two surfaces, the first surface defined in the contact property assignment is treated as a slave surface and the second surface as its corresponding master surface. For contact interactions between the cohesive surfaces and other parts of the general contact domain, the default contact formulation (balanced master-slave) is applicable, unless a nondefault general contact formulation has been defined (see "Contact formulation for general contact in Abaqus/Explicit," Section 36.2.1). The surface-based cohesive behavior is available only for node-to-face contact interactions; it is not available for edge-to-edge interactions. Hence, it is not possible to define surface-based cohesion between edges of beam and truss elements.

Care should be exercised when cohesive behavior is used in conjunction with stacked conventional shell elements. Depending on the load case, the specialized contact formulation may lead to approximate normal contact forces, which in turn may induce approximate transverse shear behavior in the stacked shells that affect the bending behavior of the stack. Continuum shells should be used instead of conventional shells in such modeling scenarios.

Resolving initial overclosures and gaps in Abaqus/Explicit

In many debonding applications using cohesive surfaces, it may be desirable to begin the analysis with the surfaces just touching each other. This requires the resolution of initial overclosures and gaps between the surfaces at the start of the analysis to ensure that the slave nodes are precisely in contact with the master surface. In Abaqus/Explicit small initial overclosures are set to zero by default. To resolve large initial overclosures or to close initial gaps between the surfaces, an appropriate contact clearance specification may be defined, as explained in "Controlling initial contact status for general contact in Abaqus/Explicit," Section 34.4.4. Since a pure-master slave formulation is enforced for cohesive surfaces, only nodes of the slave surface will undergo strain-free corrections to resolve any initial overclosures or gaps with their master facets; the nodes of the master facets will not be moved.

Defining cohesive behavior in Abaqus/Standard

Cohesive behavior in Abaqus/Standard is defined as part of the surface interaction properties that are assigned to a contact pair. Cohesive behavior cannot be assigned to contact pairs using the finite sliding, surface-to-surface formulation (see "Contact formulations in Abaqus/Standard," Section 36.1.1).

Input File Usage: Use the following options to define cohesive behavior between the surfaces in a contact pair:

∗SURFACE INTERACTION, NAME=*name*
∗COHESIVE BEHAVIOR
∗CONTACT PAIR, INTERACTION=*name*
surface1, *surface2*

Abaqus/CAE Usage: Use the following option to define cohesive behavior between two surfaces:

Interaction module: contact property editor: **Mechanical→Cohesive Behavior**

Use the following option to define surface-to-surface contact between two surfaces:

Interaction module: interaction editor: **Surface-to-surface contact (Standard)**: **Bonding** tabbed page: specify **Contact interaction property**

Resolving initial overclosures and gaps in Abaqus/Standard

As discussed above, it is often desirable in debonding applications for the cohesive surfaces to begin the analysis just touching each other. Abaqus/Standard offers some tools for adjusting slave nodes in a contact pair so that they precisely contact the master surface, thereby eliminating initial overclosures and gaps. If nodes are not adjusted, even an extremely small initial gap will cause the contact constraints to be initialized to inactive and, thus, not cohered. These tools are described in "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5.

Controlling the set of cohered nodes

By default, cohesive constraint forces can potentially act on all nodes of the surfaces for which cohesive behavior is defined. Slave nodes that are initially contacting the master surface can experience cohesive forces at the start of the analysis, and slave nodes that are not initially contacting the master surface can experience cohesive forces if they contact the master surface during the analysis. There may, however, be situations where it is desirable to enforce cohesive behavior only for portions of surfaces that are contacting at the start of the analysis.

Restricting cohesive behavior to initially contacting nodes

As part of the cohesive behavior definition, you can indicate that only those nodes that are in contact with the master surface at the start of the step should experience cohesive forces. Any new contacts that occur during the step will not experience cohesive constraint forces; they will be modeled only as compressive contact.

Input File Usage: ∗COHESIVE BEHAVIOR, ELIGIBILITY=ORIGINAL CONTACTS

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical→Cohesive Behavior**: **Only slave nodes initially in contact**

Restricting cohesive behavior to specified nodes

In Abaqus/Standard you can specify a subset of initially slave nodes that should experience cohesive forces. Strain-free adjustments will be made for those nodes initially not in contact but specified in the node set. All slave nodes outside of this set (including those that are initially contacting the master surface) will experience only compressive contact forces over the course of the analysis. This method is particularly useful for modeling crack propagation along an existing fault line.

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 slave nodes in sub-set**

Interaction of traction-separation behavior with compressive and friction behavior

In the contact normal direction, the pressure overclosure relationship governing the compressive behavior between the surfaces does not interact with the cohesive behavior, since they each describe the interaction between the surfaces in a different contact regime. The pressure overclosure relationship governs the behavior only when a slave node is "closed" (i.e., it is in contact with the master surface); the cohesive behavior contributes to the contact normal stress only when a slave node is "open" (i.e., not in contact). In the case of "sticky" cohesive behavior—where the two surfaces are not initially in contact—cohesive effects are activated in the increment after the slave node status changes from open to closed.

In the shear direction, if the cohesive stiffness is undamaged, it is assumed that the cohesive model is active and the friction model is dormant. Any tangential slip is assumed to be purely elastic in nature and is resisted by the cohesive strength of the bond, resulting in shear forces. If damage has been defined, the cohesive contribution to the shear stresses starts degrading with damage evolution. Once the cohesive stiffness starts degrading, the friction model activates and begins contributing to the shear stresses. The elastic stick stiffness of the friction model is ramped up in proportion to the degradation of the elastic cohesive stiffness. Prior to the ultimate failure of the cohesive bond, and following the initiation of the degradation of the cohesive bond, the shear stress is a combination of the cohesive contribution and the contribution from the friction model. Once maximum degradation has been reached, the cohesive contribution to the shear stresses is zero, and the only contribution to the shear stresses is from the friction model.

Applying cohesive material concepts to surface-based cohesive behavior

The formulae and laws that govern cohesive surface behavior are very similar to those used for cohesive elements with traction-separation constitutive behavior ("Defining the constitutive response of cohesive

elements using a traction-separation description," Section 31.5.6). The similarities extend to the linear elastic traction-separation model, damage initiation criteria, and damage evolution laws.

However, it is important to recognize that damage in surface-based cohesive behavior is an interaction property, not a material property. Concepts of strain and displacement (used in behavior model formulae for cohesive elements) are reinterpreted as contact separations; contact separations are the relative displacements between the nodes on the slave surface and their corresponding projection points on the master surface along the contact normal and shear directions. Stresses are defined for surface-based cohesive behavior as the cohesive forces acting along the contact normal and shear directions divided by the current area at each contact point.

The specifics of the surface-based cohesive behavior model are discussed in the sections that follow.

Linear elastic traction-separation behavior

The available traction-separation model in Abaqus assumes initially linear elastic behavior (see "Defining elasticity in terms of tractions and separations for cohesive elements" in "Linear elastic behavior," Section 21.2.1) followed by the initiation and evolution of damage. The elastic behavior is written in terms of an elastic constitutive matrix that relates the normal and shear stresses to the normal and shear separations across the interface.

The nominal traction stress vector, $\mathbf{t}$, consists of three components (two components in two-dimensional problems): t_n, t_s, and (in three-dimensional problems) t_t, which represent the normal (along the local 3-direction in three dimensions and along the local 2-direction in two dimensions) and the two shear tractions (along the local 1- and 2-directions in three dimensions and along the local 1-direction in two dimensions), respectively. The corresponding separations are denoted by δ_n, δ_s, and δ_t. The elastic behavior can then be written as

$$\mathbf{t} = \begin{Bmatrix} t_n \\ t_s \\ t_t \end{Bmatrix} = \begin{bmatrix} K_{nn} & K_{ns} & K_{nt} \\ K_{ns} & K_{ss} & K_{st} \\ K_{nt} & K_{st} & K_{tt} \end{bmatrix} \begin{Bmatrix} \delta_n \\ \delta_s \\ \delta_t \end{Bmatrix} = \mathbf{K}\boldsymbol{\delta}.$$

Uncoupled traction-separation behavior

The simplest specification of cohesive behavior generates contact penalties that enforce the cohesive constraint in both normal and tangential directions. By default, the normal and tangential stiffness components will not be coupled: pure normal separation by itself does not give rise to cohesive forces in the shear directions, and pure shear slip with zero normal separation does not give rise to any cohesive forces in the normal direction.

For uncoupled traction-separation behavior, the terms K_{nn}, K_{ss}, and K_{tt} must be defined, as well as any dependencies on temperature or field variables. If these terms are not defined, Abaqus uses default contact penalties to model the traction-separation behavior.

Input File Usage: *COHESIVE BEHAVIOR, TYPE=UNCOUPLED (default)

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical→Cohesive Behavior**: **Specify stiffness coefficients**: **Uncoupled**

Coupled traction-separation behavior

In its full generality, the elasticity matrix provides fully coupled behavior between all components of the traction vector and separation vector and can depend on temperature and/or field variables. All terms in the matrix must be defined for coupled traction-separation behavior.

Input File Usage: *COHESIVE BEHAVIOR, TYPE=COUPLED

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical→Cohesive Behavior**: **Specify stiffness coefficients**: **Coupled**

Cohesive behavior in the normal or shear direction only

To restrict the cohesive constraint to act along the contact normal direction only, define uncoupled cohesive behavior and specify zero values for the shear stiffness components, K_{ss} and K_{tt}. Alternatively, if only tangential cohesive constraints are to be enforced, the normal stiffness term, K_{nn}, can be set to zero, in which case the normal "separations" will not be constrained. Normal compressive forces are resisted as per the usual contact behavior.

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 35.1.10–1 shows a typical traction-separation response with a failure mechanism. If the damage initiation criterion is specified without a corresponding damage evolution model, Abaqus 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.

Damage of the traction-separation response for cohesive surfaces is defined within the same general framework used for conventional materials (see "Progressive damage and failure," Section 23.1.1), 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

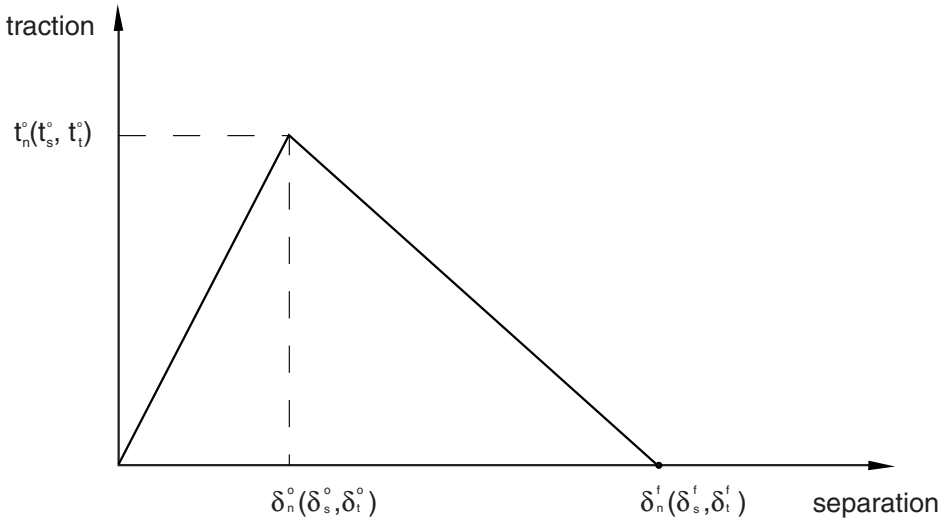


Figure 35.1.10–1 Typical traction-separation response.

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 (i.e., 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 (i.e., 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**

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," Section 23.2.3. 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 "Applying cohesive material concepts to surface-based cohesive behavior").

Mixed-mode definition

The relative proportions of normal and shear separations at a contact point define the mode mix at the point. Abaqus uses two measures of mode mix, one based on energies and the other 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 $m_3/(m_2 + m_3)(= G_t/G_S)$.

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 35.1.10–2.

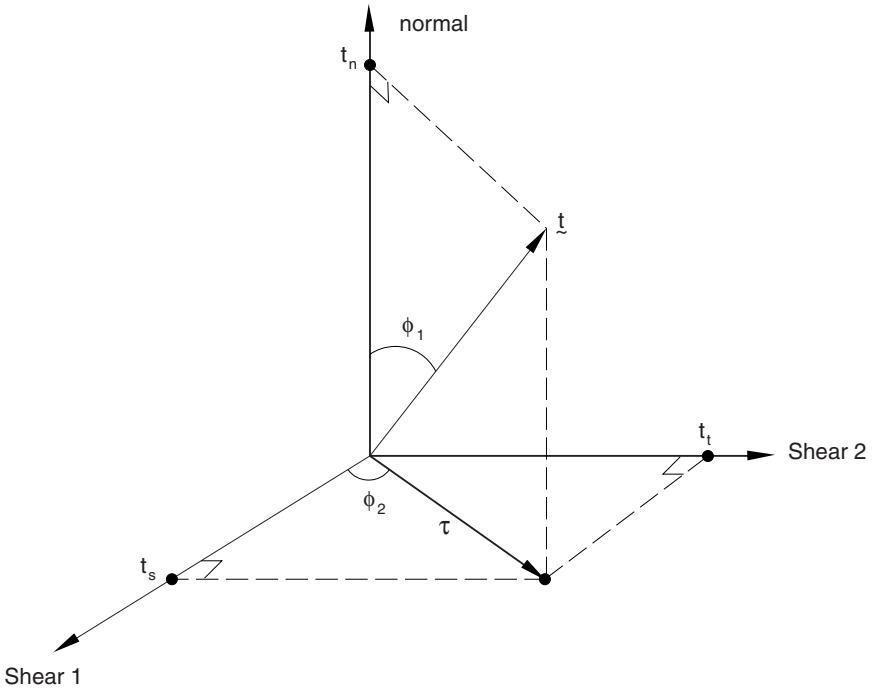


Figure 35.1.10–2 Mode-mix measures based on traction.

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.

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 35.1.10–3). 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, and/or field variables.

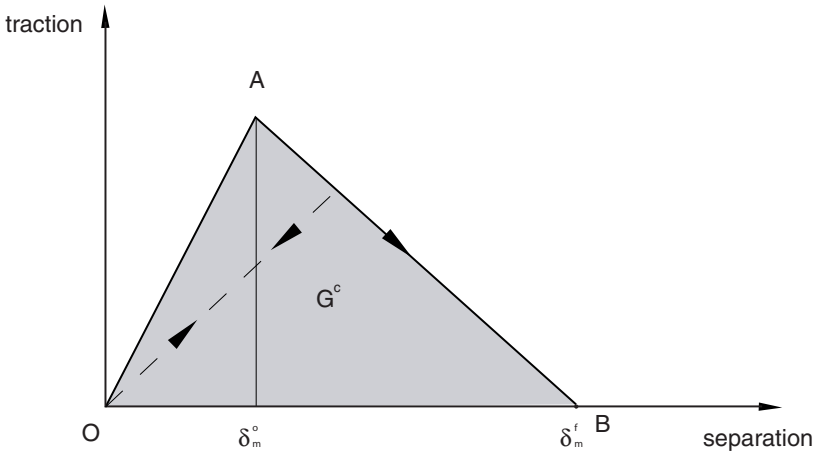


Figure 35.1.10–3 Linear damage evolution.

Figure 35.1.10–4 is a schematic representation of the dependence of damage initiation and evolution on the mode mix for a traction-separation response with isotropic shear behavior. The figure shows the traction on the vertical axis and the magnitudes of the normal and the shear separations along the two horizontal axes. The unshaded triangles in the two vertical coordinate planes represent the response under pure normal and pure shear 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.

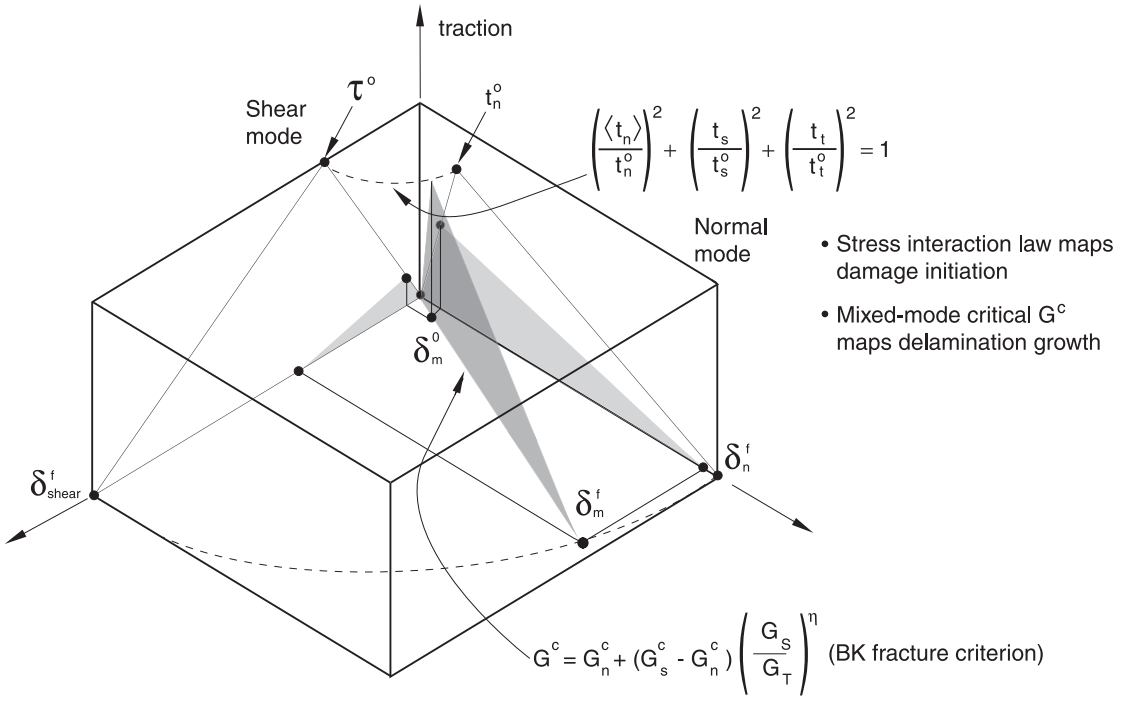


Figure 35.1.10–4 Illustration of mixed-mode response in cohesive interactions.

Unloading subsequent to damage initiation is always assumed to occur linearly toward the origin of the traction-separation plane, as shown in Figure 35.1.10–3. Reloading subsequent to unloading also occurs along the same linear path until the softening envelope (line AB) is reached. Once the softening envelope is reached, further reloading follows this envelope as indicated by the arrow in Figure 35.1.10–3.

Input File Usage: Use the following option to use the mode-mix definition based on energies:

*DAMAGE EVOLUTION, MODE MIX RATIO=ENERGY

Use the following option to use the mode-mix definition based on tractions:

*DAMAGE EVOLUTION, MODE MIX RATIO=TRACTION

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical→Damage**: **Evolution** tabbed page: toggle on **Specify mixed-mode behavior**: **Mode mix ratio**: **Energy** or **Traction**

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 35.1.10–3) 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 35.1.10–3) Abaqus uses an evolution of the damage variable, D, that reduces (in the case of damage evolution under a constant mode mix, temperature, and field variables) to the following expression:

$$D = \frac{\delta_m^f(\delta_m^{\max} - \delta_m^o)}{\delta_m^{\max}(\delta_m^f - \delta_m^o)}.$$

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 35.1.10–5) Abaqus uses an evolution of the damage variable, D, that reduces (in the case of damage evolution under a constant mode mix, temperature, and field variables) to

$$D = 1 - \left\{\frac{\delta_m^o}{\delta_m^{\max}}\right\}\left\{1 - \frac{1 - \exp(-\alpha(\frac{\delta_m^{\max} - \delta_m^o}{\delta_m^f - \delta_m^o}))}{1 - \exp(-\alpha)}\right\}.$$

In the expression above α is a non-dimensional parameter that defines the rate of damage evolution and $\exp(x)$ is the exponential function.

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**

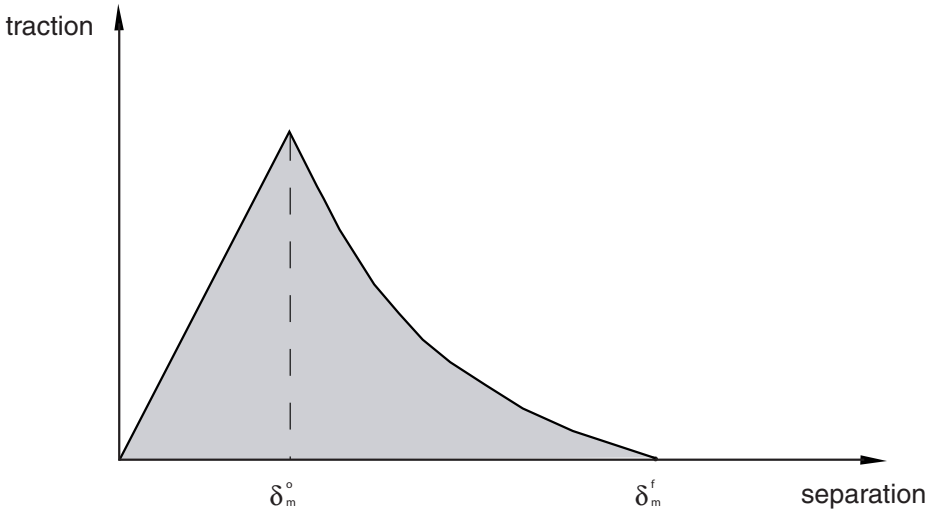


Figure 35.1.10–5 Exponential damage evolution.

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 35.1.10–3). 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

$$G_n^C + (G_s^C - G_n^C)\left\{\frac{G_S}{G_T}\right\}^\eta = G^C,$$

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 35.1.10–3) Abaqus uses an evolution of the damage variable, D, that reduces to

$$D = \frac{\delta_m^f(\delta_m^{\max} - \delta_m^o)}{\delta_m^{\max}(\delta_m^f - \delta_m^o)},$$

where $\delta_m^f = 2G^C/T_{\text{eff}}^o$ with T_{eff}^o as the effective traction at damage initiation. $\delta_m^{\max}$ refers to the maximum value of the effective 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

$$D = \int_{\delta_m^o}^{\delta_m^f} \frac{T_{\text{eff}}\, d\delta}{G^C - G_o}.$$

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 35.1.10–5.

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 35.1.10–6 shows a schematic of the fracture energy versus mode-mix behavior. The limiting cases of pure normal and pure shear separations in the first and second shear directions are denoted in Figure 35.1.10–6 by G_n^C, G_s^C, and G_t^C, respectively. The lines labeled "Modes n-s," "Modes n-t," and "Modes s-t" show the transition in behavior between the pure normal and the pure shear in the first direction, pure normal and pure shear in the second direction, and pure shears in the first and second directions, respectively. In general, G^C must be specified as a function of (m_2+m_3) at various fixed values of $[m_3/(m_2+m_3)]$. In the discussion that follows we refer to a data set of G^C versus (m_2+m_3) corresponding to a fixed $[m_3/(m_2+m_3)]$ as a "data block." The following guidelines are useful in defining the fracture energy as a function of the mode mix:

- For a two-dimensional problem G^C needs to be defined as a function of m_2 ($m_3 = 0$ in this case) only. The data column corresponding to $[m_3/(m_2+m_3)]$ must be left blank. Hence, essentially only one "data block" is needed.
- For a three-dimensional problem with isotropic shear response, the shear behavior is defined by the sum (m_2+m_3) and not by the individual values of m_2 and m_3. Therefore, in this case a single "data block" (the "data block" for $[m_3/(m_2+m_3)] = 0$) also suffices to define the fracture energy as a function of the mode mix.
- In the most general case of three-dimensional problems with anisotropic shear behavior, several "data blocks" would be needed. As discussed earlier, each "data block" would contain G^C versus (m_2+m_3) at a fixed value of $[m_3/(m_2+m_3)]$. In each "data block" (m_2+m_3) can vary between 0 and 1.0. The case $(m_2+m_3) = 0$ (the first data point in any "data block"), which corresponds to a purely normal mode, can never be achieved when $[m_3/(m_2+m_3)] \neq 0$ (i.e., the only valid point on line OB in Figure 35.1.10–6 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

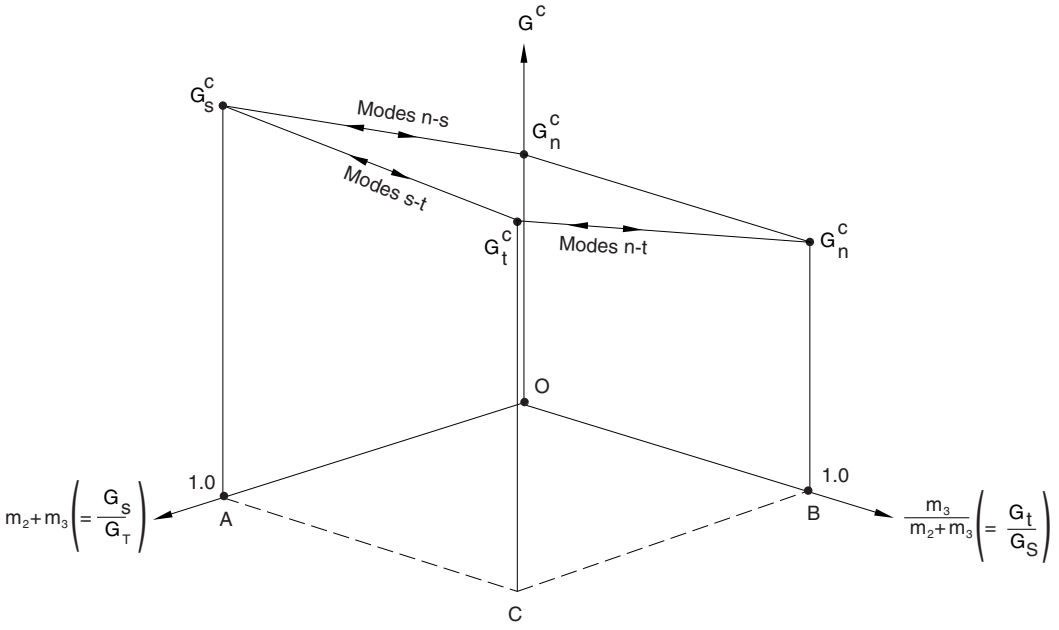


Figure 35.1.10–6 Fracture energy as a function of mode mix.

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 35.1.10–2).

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 available using output variable ALLVD.

Input File Usage: *DAMAGE STABILIZATION

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical→Damage**: **Stabilization** tabbed page: **Viscosity coefficient**

Post-failure behavior

Two types of post-failure behavior can be specified to define the cohesive behavior at a node on the slave surface after the maximum degradation value, $D_{max} = 1.0$, has been reached at the node.

By default, once fully degraded, normal contact behavior is enforced at the node and no further cohesive constraints are enforced. If the slave node re-enters contact, penetrations will give rise to compressive contact stresses, and frictional stresses will be applied in the shear directions according to the prescribed friction model, if any. Separations can occur without giving rise to any cohesive stresses.

In some situations it may be desirable to enforce cohesive behavior again if a slave node re-enters contact, even after maximum degradation has been reached. For cohesive behavior allowing repeated contacts, the overall damage variable will be re-initialized to zero when a failed slave node re-enters contact. Subsequently, normal separations may give rise to tensile cohesive stresses, and shear separations may give rise to tangential cohesive stresses in accordance with the type of cohesive behavior defined. Further loading can again cause the cohesive stresses to undergo progressive damage, degrade, and fail.

Input File Usage: Use the following option to enforce cohesive behavior subsequent to maximum degradation:

*COHESIVE BEHAVIOR, REPEATED CONTACTS

Abaqus/CAE Usage: Interaction module: contact property editor: **Mechanical→Cohesive Behavior**: **Allow cohesive behavior during repeated post-failure contacts**

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," Section 11.4.3.

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
```

Cohesive surfaces versus cohesive elements

As described above, the formulation used for surface-based cohesive behavior is very similar to that for cohesive elements with traction-separation response. However, certain differences exist.

Interface thickness effects are never considered for cohesive surfaces; in cohesive elements with traction-separation response, thickness effects can be incorporated by either specifying a nonzero thickness for the interface or by requiring the initial constitutive thickness to be determined from the nodal coordinates of the cohesive elements. Since thickness effects are not considered for cohesive surfaces, material properties used to describe the constitutive response for traction-separation cohesive elements with thickness effects may not be directly reusable for cohesive surfaces.

For cohesive surfaces the cohesive constraint is enforced at each slave node; in cohesive elements the cohesive constraints are calculated at the material points (for the locations of material points in cohesive elements, see "Two-dimensional cohesive element library," Section 31.5.8, and "Three-dimensional cohesive element library," Section 31.5.9). Hence for cohesive surfaces, refining the slave surface as compared to the master surface will likely lead to improved constraint satisfaction and more accurate results.

Output

In addition to the standard output identifiers available in Abaqus ("Abaqus/Standard output variable identifiers," Section 4.2.1, and "Abaqus/Explicit output variable identifiers," Section 4.2.2), 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. It is evaluated as $\max\{\frac{\langle t_n \rangle}{t_n^o}, \frac{t_s}{t_s^o}, \frac{t_t}{t_t^o}\}$.

CSMAXUCRT — This variable indicates whether the maximum separation damage initiation criterion has been satisfied at a contact point. It is evaluated as $\max\{\frac{\langle \delta_n \rangle}{\delta_n^o}, \frac{\delta_s}{\delta_s^o}, \frac{\delta_t}{\delta_t^o}\}$.

CSQUADSCRT — This variable indicates whether the quadratic contact stress damage initiation criterion has been satisfied at a contact point. It is evaluated as $(\frac{\langle t_n \rangle}{t_n^o})^2 + (\frac{t_s}{t_s^o})^2 + (\frac{t_t}{t_t^o})^2$.

CSQUADUCRT — This variable indicates whether the quadratic separation damage initiation criterion has been satisfied at a contact point. It is evaluated as $(\frac{\langle \delta_n \rangle}{\delta_n^o})^2 + (\frac{\delta_s}{\delta_s^o})^2 + (\frac{\delta_t}{\delta_t^o})^2$.

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. If damage evolution is specified for this criterion, the maximum value of this variable does not exceed 1.0.

Additional 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.

35.2 Thermal contact properties

- “Thermal contact properties,” Section 35.2.1

35.2.1 THERMAL CONTACT PROPERTIES

Products: Abaqus/Standard Abaqus/Explicit Abaqus/CAE

References

- "Contact interaction analysis: overview," Section 34.1.1
- "User-defined interfacial constitutive behavior," Section 35.1.6
- "GAPCON," Section 1.1.10 of the Abaqus User Subroutines Reference Manual
- *GAP
- *GAP CONDUCTANCE
- *GAP HEAT GENERATION
- *GAP RADIATION
- *INTERFACE
- *SURFACE INTERACTION
- "Creating interaction properties," Section 15.12.2 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Thermal interaction at the surface of a body:

- can be included in heat transfer problems ("Uncoupled heat transfer analysis," Section 6.5.2; "Fully coupled thermal-stress analysis," Section 6.5.4; "Fully coupled thermal-electrical-structural analysis," Section 6.7.4; and "Coupled thermal-electrical analysis," Section 6.7.3);
- 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 thermal-mechanical 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," Section 39.1.1. The thermal contact property models described here are for bodies in close proximity or in contact. For these problems gap radiation may be more efficient and robust than cavity radiation.

Including thermal properties in a contact property definition

All of the thermal properties discussed in this section—gap conductance, gap radiation, and gap heat generation—can be included 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.

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," Section 35.1.6).

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.

Thermal contact considerations in Abaqus/Explicit

Gap conductance and gap radiation are enforced in Abaqus/Explicit with an explicit algorithm analogous to the penalty method for mechanical contact interaction. Therefore, gap 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," Section 6.5.4). Extremely large values of gap conductance or gap radiation can result in a decrease in the stable time increment, which will be 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 may 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 may cause discontinuity in the contact pressure: for pressure-dependent gap conductance, the gap conduction coefficient will be 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 will depend 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 for general contact works analogously to thermal contact for contact pairs. Gap conductance, gap radiation, and gap heat generation can all be specified and incorporated in general contact definitions through contact property assignments. As discussed above, large values of gap conductance or gap radiation can result in performance degradation, particularly since more surfaces are typically involved in general contact than in contact pairs. Thermal contact properties cannot be specified for general contact involving edge-to-edge contact or Eulerian elements. Thermal contact properties are ignored when shell elements are used to define surfaces involved in a contact pair definition. In these cases general contact should be used.

Modeling 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 gap conductance. Point A is a node on the slave surface; and point B is the location on the master surface contacting the slave node or, if the surfaces are not in contact, the location on the master surface with a surface normal that intersects the slave node.

You can define k directly or, in Abaqus/Standard, in user subroutine **GAPCON**.

Defining the gap conductance directly

When defining k directly, define it as

$$k = k(\bar{\theta}, d, p, \overline{|\dot{m}|}, \bar{f}_\gamma),$$

where

d	is the clearance between A and B,
p	is the contact pressure transmitted across the interface between A and B,
$\bar{\theta} = \frac{1}{2}(\theta_A + \theta_B)$	is the average of the surface temperatures at A and B,
$\overline{\lvert\dot{m}\rvert} = \frac{1}{2}(\lvert\dot{m}\rvert_A + \lvert\dot{m}\rvert_B)$	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).

$\bar{f}_\gamma = \frac{1}{2}(f_\gamma^A + f_\gamma^B)$ is the average of any predefined field variables at ***A*** and ***B***, and

Defining gap conductance as a function of clearance

You can create a table of data defining the dependence of k on the variables listed above. The default in Abaqus is to make k a function of the clearance d. When k is a function of gap clearance, d, the tabular data must start at zero clearance (closed gap) and define k as d increases. At least two pairs of $k-d$ points must be given to define k as a function of the clearance. The value of k drops to zero immediately after the last data point, so there is no heat conductance when the clearance is greater than the value corresponding to the last data point. If gap conductance is not also defined as a function of contact pressure, k will remain constant at the zero clearance value for all pressures, as shown in Figure 35.2.1–1(a).

Input File Usage: *GAP CONDUCTANCE
k, d, $\bar{\theta}$

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal→Thermal Conductance**: **Definition: Tabular**, **Use only clearance-dependency data**

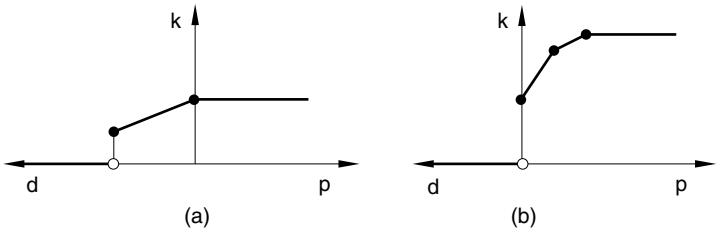


Figure 35.2.1–1 Examples of input data to define the gap conductance as a function of clearance or contact pressure.

Defining gap conductance as a function of contact pressure

You can define k as a function of the contact pressure, p. 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 force, the data point with the most negative pressure) and define k as p increases. The value of k remains constant for contact pressures outside of the interval defined by the data points. If gap conductance is not also defined as a function of clearance, k is zero for all positive values of clearance and discontinuous at zero clearance, as shown in Figure 35.2.1–1(b).

Input File Usage: *GAP CONDUCTANCE, PRESSURE
k, p, $\bar{\theta}$

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal→Thermal Conductance**: **Definition: Tabular**, **Use only pressure-dependency data**

Gap conductance as a function of both clearance and contact pressure

$\boldsymbol{k}$ can depend on both clearance and pressure. A discontinuity in $\boldsymbol{k}$ is allowed at $d = 0$ and $p = 0$. At the state of zero clearance and zero pressure the value of $\boldsymbol{k}$ corresponding to the zero pressure data point is used, as shown in Figure 35.2.1–2(a).

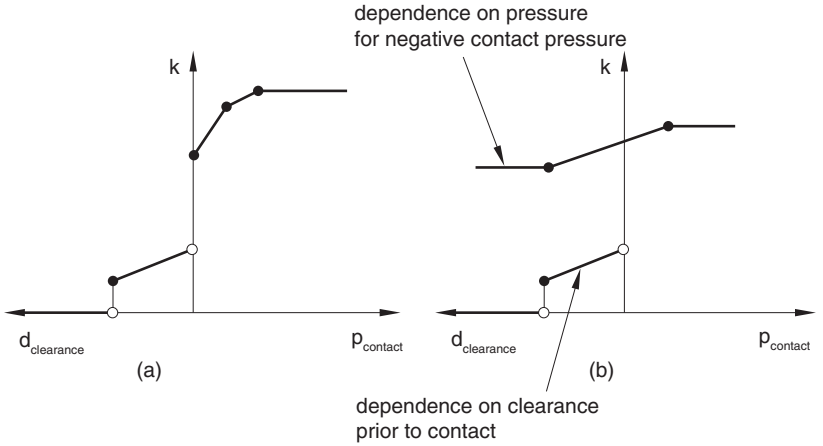


Figure 35.2.1–2 Examples of input data to define the gap conductance as a function of both clearance and contact pressure.

In the case of no-separation contact, once contact occurs the conductance is always evaluated based on the portion of the curve that defines the pressure dependence. The gap conductance, $\boldsymbol{k}$, remains constant for contact pressures outside of the interval defined by the data points, as shown in Figure 35.2.1–2(b). The pressure dependence of $\boldsymbol{k}$ is extended into the negative pressure region even if no data points with negative pressure are included.

Input File Usage:

∗GAP CONDUCTANCE
k, $\boldsymbol{d}$, $\bar{\theta}$
∗GAP CONDUCTANCE, PRESSURE
k, $\boldsymbol{p}$, $\bar{\theta}$

For example, the following input defines $k = 20$ for the zero clearance data point and $k = 50$ for the zero pressure data point:

```
*SURFACE INTERACTION, NAME=name
*GAP CONDUCTANCE
20.0, 0.0
10.0, 0.1
…
*GAP CONDUCTANCE, PRESSURE
50.0, 0.0
```

```
65.0, 100.0
70.0, 250.0
...
```

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal→Thermal Conductance**: **Definition: Tabular**, **Use both clearance- and pressure-dependency data**

Using gap 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" in "Uncoupled heat transfer analysis," Section 6.5.2) 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 will be half the magnitude specified.

Input File Usage: *GAP CONDUCTANCE
k, d, $\bar{\theta}$, $\overline{|\dot{m}|}$

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 gap conductance to be a function of predefined field variables

In addition to the dependencies mentioned previously, the gap conductance can be dependent on any number of predefined field variables, $\bar{f}_\gamma$. To make the gap conductance depend on field variables, at least two data points are required for each field variable value.

Input File Usage: *GAP CONDUCTANCE, DEPENDENCIES=n
k, d, $\bar{\theta}$, $\overline{|\dot{m}|}$, $\bar{f}_\gamma$

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal→Thermal Conductance**: **Definition: Tabular**, **Clearance Dependency** and/or **Pressure Dependency**, **Number of field variables:** n

Defining the gap conductance using user subroutine GAPCON

In Abaqus/Standard k can be defined in user subroutine **GAPCON**. In this case there is greater flexibility in specifying the dependencies of k. It is no longer necessary to define k as a function of the average of the two surface's temperatures, mass flow rates, or field variables.

$$k = k(d, p, \theta_A, \theta_B, |\dot{m}|_A, |\dot{m}|_B, f_\gamma^A, f_\gamma^B).$$

Input File Usage: *GAP CONDUCTANCE, USER

Abaqus/CAE Usage: Interaction module: contact property editor: **Thermal→Thermal Conductance**: **Definition: User-defined**

Defining the gap 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 may improve the convergence rate in the analysis (see "Procedures: overview," Section 6.1.1).

Temperature and field-variable dependence of gap 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 gap conductance, even in cases where the properties are also clearance dependent.

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 master surface normal (see "Contact formulations in Abaqus/Standard," Section 36.1.1; "Defining contact pairs in Abaqus/Explicit," Section 34.5.1; and "Surfaces: overview," Section 2.3.1). 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," Section 39.1.1).

Radiative heat transfer is defined as a function of clearance between the surfaces through the effective viewfactor. 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

$$q = C[(\theta_A - \theta^Z)^4 - (\theta_B - \theta^Z)^4],$$

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 viewfactor, which corresponds to viewing the master surface from the slave surface.

The viewfactor F must be defined as a function of the clearance, d, and should have a value between 0.0 and 1.0. At least two pairs of $F - d$ points are required to define the viewfactor, and the tabular data must start at zero clearance (closed gap) and define the viewfactor 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 35.2.1–3).

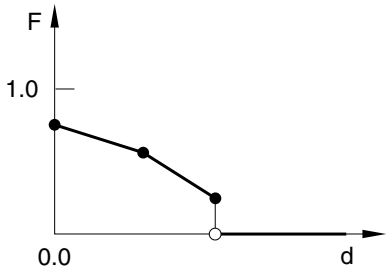


Figure 35.2.1–3 Example of input data to define the viewfactor as a function of clearance.

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 master surface:** ϵ_A, **Emissivity of slave surface:** ϵ_B, **Viewfactor** 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 may improve the convergence rate in Abaqus/Standard (see "Procedures: overview," Section 6.1.1).

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 (slave) surface of the contact pair. $f = 0.0$ indicates that all of the generated heat flows into the opposite (master) 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.

If user subroutine **UINTER**, **VUINTER**, or **VUINTERACTION** is used to define the interfacial constitutive behavior, all gap heat generation effects will be 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 thermal-mechanical 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 slave surface is q_A, and the heat into the master 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," Section 6.7.3) and a fully coupled thermal-electrical-structural analysis (see "Fully coupled thermal-electrical-structural analysis," Section 6.7.4), the rate of electrical energy dissipated by electric current flowing across the interface is

$$P_{ec} = J\,(\varphi_A - \varphi_B),$$

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 slave surface is q_A, and the heat into the master 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 slave surface. In Abaqus/Explicit these variables can be output for master and slave 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" in "Output to the data and results files," Section 4.1.2, and "Surface output in Abaqus/Standard and Abaqus/Explicit" in "Output to the output database," Section 4.1.3, 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).

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" in "Output to the data and results files," Section 4.1.2, and "Element output" in "Output to the output database," Section 4.1.3, for details). The only nonzero component will be 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 master surface side of the element (see "Gap contact elements," Section 38.2.1, 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," Section 2.4.1. 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," Section 6.5.4) in Abaqus/Standard:

- No heat flow will occur across a contact pair involving a node-based surface.
- No heat generation will occur 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 "Heat transfer analysis procedures: overview," Section 6.5.1).

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," Section 2.3.3).

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; i.e., 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.

35.3 Electrical contact properties

- “Electrical contact properties,” Section 35.3.1

35.3.1 ELECTRICAL CONTACT PROPERTIES

Product: Abaqus/Standard

References

- "Contact interaction analysis: overview," Section 34.1.1
- "Thermal contact properties," Section 35.2.1
- "GAPELECTR," Section 1.1.11 of the Abaqus User Subroutines Reference Manual
- *GAP ELECTRICAL CONDUCTANCE
- *SURFACE INTERACTION

Overview

Electrical conduction between two bodies:

- is proportional to the difference in electric potentials across the interface;
- is a function of the clearance between the surfaces;
- can be a function of contact pressure;
- can be a function of surface temperatures and/or predefined field variables on the surfaces; and
- can generate heat at the interface.

See "Coupled thermal-electrical analysis," Section 6.7.3, and "Fully coupled thermal-electrical-structural analysis," Section 6.7.4, for details on coupled thermal-electrical and coupled thermal-electrical-structural analyses.

Including gap electrical conductance properties in a contact property definition

You can include electrical conductance properties in a contact property definition for surface-based contact.

Input File Usage: Use both of the following options:

*SURFACE INTERACTION, NAME=*name*
*GAP ELECTRICAL CONDUCTANCE

Modeling electrical conductance between surfaces

Abaqus/Standard models the electrical current flowing between two surfaces as

$$J = \sigma_g (\varphi_A - \varphi_B),$$

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 gap electrical conductance. Point A corresponds to a node on the slave surface of the contact pair. Point B is the point of the master surface in contact with point A.

You can provide the electrical conductance directly or in user subroutine **GAPELECTR**.

Defining σ_g directly

When the gap electrical conductance is defined directly, Abaqus/Standard assumes that

$$\sigma_g = \sigma_g(\bar{\theta}, d, p, \bar{f}^\alpha),$$

where

$\bar{\theta} = \frac{1}{2}(\theta_A + \theta_B)$	is the average of the surface temperatures at A and B,
d	is the clearance between A and B,
p	is the contact pressure transmitted across the interface between A and B, and
$\bar{f}^\alpha = \frac{1}{2}(f_A^\alpha + f_B^\alpha)$	is the average of any predefined field variables at A and B.

Defining gap electrical conductance as a function of clearance

You can create a table of data defining the dependence of σ_g on the variables listed above. The default in Abaqus is to make σ_g a function of the clearance, d. When σ_g is a function of gap clearance, d, the tabular data must start at zero clearance (closed gap) and define σ_g as a function of the clearance. The value of σ_g remains constant for clearances outside of the interval defined by the data points. If gap electrical conductance is not also defined as a function of contact pressure, σ_g will remain constant at the zero clearance value for all pressures, as shown in Figure 35.3.1–1(a).

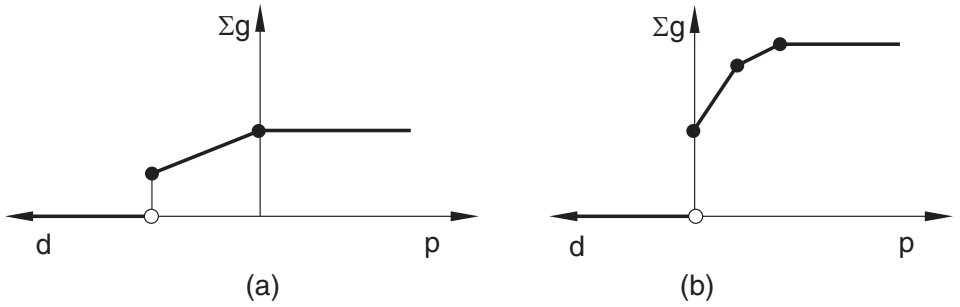


Figure 35.3.1–1 Examples of defining the gap electrical conductance as a function of clearance (a) or contact pressure (b).

Input File Usage: *GAP ELECTRICAL CONDUCTANCE
σ_g, d, θ

Defining gap electrical conductance as a function of contact pressure

You can define σ_g as a function of the contact pressure, p. 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 force, 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 outside of the interval defined by the data points. If gap electrical conductance is not also defined as a function of clearance, σ_g is zero for all positive values of clearance and discontinuous at zero clearance, as shown in Figure 35.3.1–1(b).

Input File Usage: *GAP ELECTRICAL CONDUCTANCE, PRESSURE
$\sigma_g,\ p,\ \theta$

Gap electrical conductance as a function of both clearance and contact pressure

You can define σ_g to depend on both clearance and pressure. A discontinuity in σ_g is allowed at $d = 0$ and $p = 0$. Once contact occurs, the conductance is always evaluated based on the portion of the curve that defines the pressure dependence. The gap electrical conductance, σ_g, remains constant for contact pressures outside of the interval defined by the data points. The pressure dependence of σ_g is extended into the negative pressure region even if no data points with negative pressure are included.

Input File Usage: Use both of the following options:

*GAP ELECTRICAL CONDUCTANCE
σ_g, d, θ
*GAP ELECTRICAL CONDUCTANCE, PRESSURE
$\sigma_g,\ p,\ \theta$

Defining gap electrical conductance to be a function of predefined field variables

The gap electrical conductance can be dependent on any number of predefined field variables, $\bar{f}^{\alpha}$. 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*

Defining σ_g 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:

$$\sigma_g = \sigma_g(\theta_A, \theta_B, d, p, f_A^{\alpha}, f_B^{\alpha}).$$

Input File Usage: *GAP ELECTRICAL CONDUCTANCE, USER

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 and a fully coupled thermal-electrical-structural 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" in "Thermal contact properties," Section 35.2.1, for details.

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 slave surface.
ECDA	ECD multiplied by the area associated with the slave node.
ECDT	Time integrated ECD.
ECDTA	Time integrated ECDA.

The values of these variables are always given at the nodes of the slave surface. They can be requested as surface output to the data, results, or output database files (see "Surface output from Abaqus/Standard" in "Output to the data and results files," Section 4.1.2, and "Surface output in Abaqus/Standard and Abaqus/Explicit" in "Output to the output database," Section 4.1.3, for details).

Contour plots of these variables can also be displayed in the Visualization module of Abaqus/CAE (Abaqus/Viewer).

35.4 Pore fluid contact properties

- “Pore fluid contact properties,” Section 35.4.1

35.4.1 PORE FLUID CONTACT PROPERTIES

Product: Abaqus/Standard

References

- "Contact interaction analysis: overview," Section 34.1.1
- *CONTACT PERMEABILITY
- *SURFACE
- *SURFACE INTERACTION
- *CONTACT PAIR

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," Section 6.8.1, for details on coupled pore fluid diffusion/stress analyses. See "Defining the constitutive response of fluid within the cohesive element gap," Section 31.5.7, 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 35.4.1–1. The fluid flow into the master and slave 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 master surface and into the slave surface.
- The other contribution (q_{gapS} for the slave surface and q_{gapM} for the master 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.

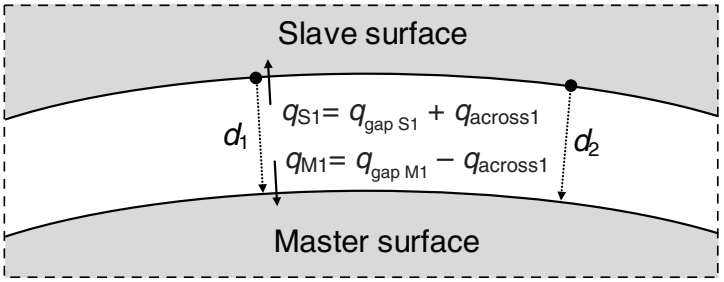


Figure 35.4.1–1 Flow patterns in the interface contact element.

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.

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 default model ensures continuity of the pore pressures on opposite sides of a contact interface 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, $\boldsymbol{k}$, 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:

$$q_{across} = k(p_A - p_B).$$

When defining $\boldsymbol{k}$ directly, define it as

$$k = k(p_{contact}, \bar{p}_{pore}, \bar{\theta}, \bar{f}_\gamma),$$

where

$p_{contact}$ is the contact pressure transmitted across the interface between $\boldsymbol{A}$ and $\boldsymbol{B}$,

$\bar{p}_{pore} = \frac{1}{2}(p_{poreA} + p_{poreB})$ is the average of the pore pressures at $\boldsymbol{A}$ and $\boldsymbol{B}$,

$\bar{\theta} = \frac{1}{2}(\theta_A + \theta_B)$ is the average of the surface temperatures at $\boldsymbol{A}$ and $\boldsymbol{B}$, and

$\bar{f}_\gamma = \frac{1}{2}(f_\gamma^A + f_\gamma^B)$ is the average of any predefined field variables at $\boldsymbol{A}$ and $\boldsymbol{B}$.

Figure 35.4.1–2 shows an example of $\boldsymbol{k}$ depending on the contact pressure. Use tabular data to specify the value of $\boldsymbol{k}$ at one or more contact pressures as $\boldsymbol{p}$ increases. The value of $\boldsymbol{k}$ remains constant for contact pressures outside of the interval defined by the data points. Once the surfaces have separated, $\boldsymbol{k}$ 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 $\boldsymbol{k}$ drops to zero.

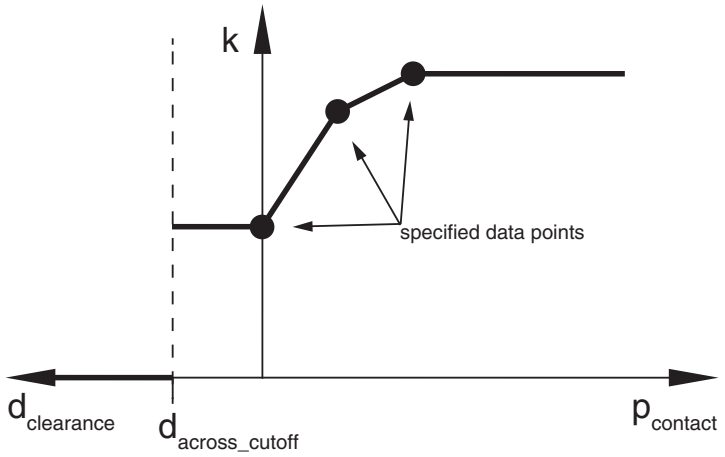


Figure 35.4.1–2 Contact-pressure-dependent contact permeability.

Input File Usage: ∗CONTACT PERMEABILITY
k, $p_{contact}$, $\bar{p}_{pore}$, $\bar{\theta}$

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, $\bar{f}_\gamma$. 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, $p_{contact}$, $\bar{p}_{pore}$, $\bar{\theta}$, $\bar{f}_\gamma$

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," Section 35.2.1, 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" in "Output to the data and results files," Section 4.1.2, and "Surface output in Abaqus/Standard and Abaqus/Explicit" in "Output to the output database," Section 4.1.3.

Abaqus/Standard provides the following output variables related to the pore fluid interaction of surfaces:

PFL	Pore volume flux per unit area leaving the slave surface.
PFLA	PFL multiplied by the area associated with the slave node.
PTL	Time integrated PFL.
PTLA	Time integrated PFLA.
TPFL	Total pore volume flux leaving the slave surface.
TPTL	Time integrated TPFL.

36. Contact Formulations and Numerical Methods

Contact formulations and numerical methods in Abaqus/Standard 36.1
Contact formulations and numerical methods in Abaqus/Explicit 36.2

36.1 Contact formulations and numerical methods in Abaqus/Standard

- "Contact formulations in Abaqus/Standard," Section 36.1.1
- "Contact constraint enforcement methods in Abaqus/Standard," Section 36.1.2
- "Smoothing contact surfaces in Abaqus/Standard," Section 36.1.3

36.1.1 CONTACT FORMULATIONS IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Surfaces: overview," Section 2.3.1
- "Defining general contact interactions in Abaqus/Standard," Section 34.2.1
- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- *CONTACT
- *CONTACT PAIR
- "Defining general contact," Section 15.13.1 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining self-contact," Section 15.13.7 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Using contact and constraint detection," Section 15.16 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Abaqus/Standard provides several contact fomulations. Each formulation is based on a choice of a contact discretization, a tracking approach, and assignment of "master" and "slave" 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 "Defining contact pairs in Abaqus/Standard," Section 34.3.1. 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," Section 5.1.2 of the Abaqus Theory Manual, and "Finite-sliding interaction between a deformable and a rigid body," Section 5.1.3 of the Abaqus Theory Manual); 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," Section 5.1.1 of the Abaqus Theory Manual).

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

General contact in Abaqus/Standard always 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 master and slave roles are assigned to surfaces automatically, although it is possible to override these default assignments. The behavior of master surfaces and slave surfaces is consistent across general contact and contact pair interactions. The specification of master and slave surfaces in a general contact domain is covered in detail in "Numerical controls for general contact in Abaqus/Standard," Section 34.2.6.

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 "slave" node on one side of a contact interface effectively interacts with a point of projection on the "master" surface on the opposite side of the contact interface (see Figure 36.1.1–1). Thus, each contact condition involves a single slave node and a group of nearby master nodes from which values are interpolated to the projection point.

Traditional node-to-surface discretization has the following characteristics:

- The slave nodes are constrained not to penetrate into the master surface; however, the nodes of the master surface can, in principle, penetrate into the slave surface (for example, see the case on the upper-right of Figure 36.1.1–2).
- The contact direction is based on the normal of the master surface.
- The only information needed for the slave surface is the location and surface area associated with each node; the direction of the slave surface normal and slave surface curvature are not relevant. Thus, the slave 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 slave and master surfaces in the region of contact constraints. Surface-to-surface discretization has the following key characteristics:

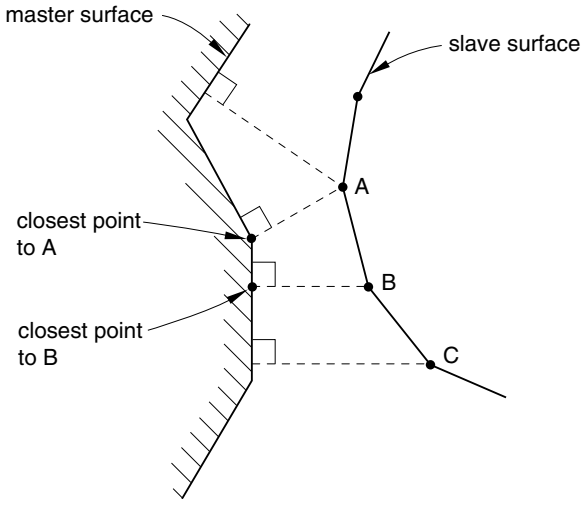


Figure 36.1.1–1 Node-to-surface contact discretization.

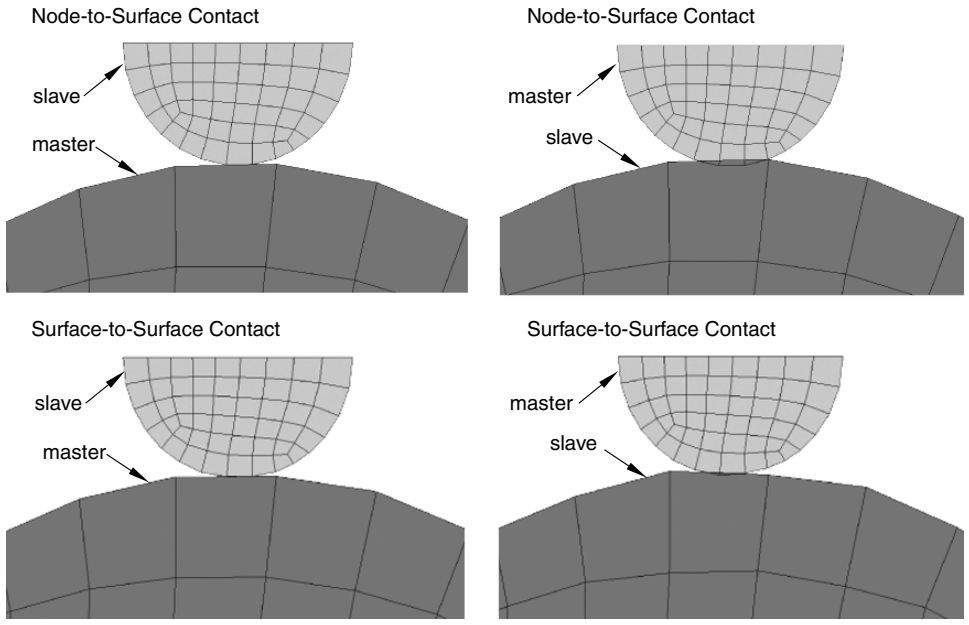


Figure 36.1.1–2 Comparison of contact enforcement for different master-slave assignments with node-to-surface and surface-to-surface contact discretizations.

- The surface-to-surface formulation enforces contact conditions in an average sense over regions nearby slave nodes rather than only at individual slave nodes. The averaging regions are approximately centered on slave nodes, so each contact constraint will predominantly consider one slave node but will also consider adjacent slave nodes. Some penetration may be observed at individual nodes; however, large, undetected penetrations of master nodes into the slave surface do not occur with this discretization. Figure 36.1.1–2 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 slave surface in the region surrounding a slave 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 36.1.1–3 shows an example of improved contact pressure accuracy with surface-to-surface contact compared to node-to-surface contact.

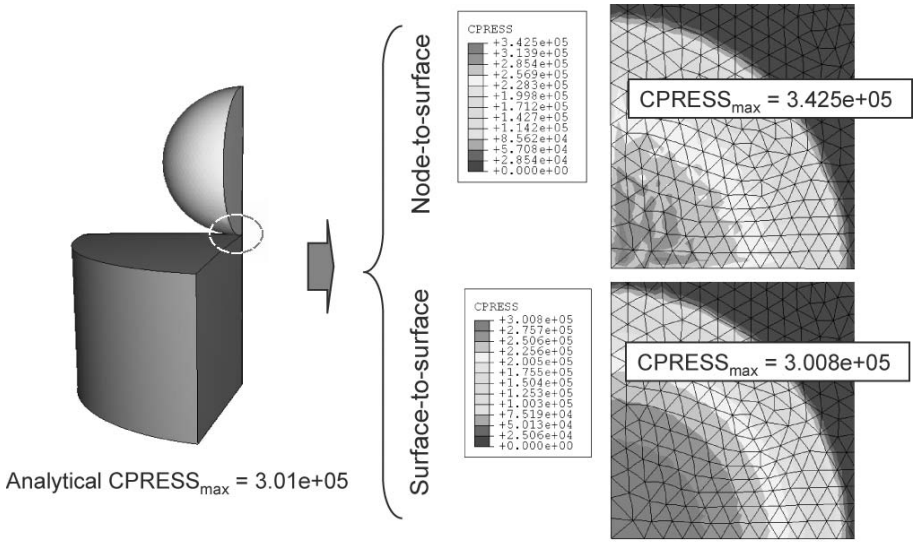


Figure 36.1.1–3 Comparison of contact pressure accuracy for node-to-surface and surface-to-surface contact discretizations.

Since node-to-surface discretization simply resists penetrations of slave nodes into the master surface, forces tend to concentrate at these slave 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 slave 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 master and slave surface designations than node-to-surface contact (see "Choosing the master and slave roles in a two-surface contact pair" below). Figure 36.1.1–4 shows a simple model involving two blocks with dissimilar mesh densities.

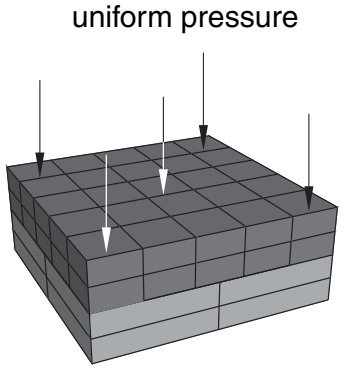


Figure 36.1.1–4 Test model for comparison of different master and slave surface designations.

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 36.1.1–1 compares the Abaqus analysis results for different contact discretizations and slave surface designations.

Table 36.1.1–1 Error (from analytical results) for various discretization/slave surface combinations.

Contact discretization	Slave 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," Section 36.1.3, 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 master surface is more refined than the slave surface.
- Multiple layers of shells are involved in contact, such that the master surface of one contact pair acts as the slave 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 slave and master 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 master 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 master and slave roles in a two-surface contact pair

Abaqus/Standard enforces the following rules related to the assignment of the master and slave roles for contact surfaces:

- Analytical rigid surfaces and rigid-element-based surfaces must always be the master surface.

- A node-based surface can act only as a slave surface and always uses node-to-surface contact.
- Slave 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," Section 2.4.1).

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 slave surface and which will be the master 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 slave surface. If that distinction cannot be made, the master 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 master and slave surface. 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 master and slave 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 master and slave 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 slave surface is much coarser than the master 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 36.1.1–2. You should also consider the solution costs associated with the various contact formulations.

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" in "Assigning surface properties for contact pairs in Abaqus/Standard," Section 34.3.2.

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.

Table 36.1.1–2 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	Slave surface only	Slave surface only	Yes[1]	Yes
Surface smoothing by default	Some smoothing of master surface	Yes for anchor points; each constraint uses flat approximation of master surface	No	No for anchor points; each constraint uses flat approximation of master surface
Default constraint enforcement method	Augmented Lagrange method for 3-D self-contact; otherwise, direct method	Direct method	Penalty method	Direct method
Ensure moment equilibrium for offset reference surfaces with friction	No	No	Yes	Yes

[1] Double-sided master 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 slave surfaces are allowed with both tracking algorithms if the master surface is not user defined.

Allowing double-sided surfaces

Doubled-sided contact surfaces based on shell-like elements are allowed to act as slave and/or master surfaces for the surface-to-surface contact formulation by default and are allowed to act as the slave surface for the node-to-surface contact formulation. For a shell-like surface to act as the master 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" in "Element-based surface

definition," Section 2.3.2, and "Orientation considerations for shell-like surfaces" in "Defining contact pairs in Abaqus/Standard," Section 34.3.1, for more information).

Surface smoothing

When using node-to-surface discretization, corners or small protrusions of a jagged master surface are allowed to penetrate the spaces between nodes in the node-based surface. It is sometimes possible for a slave node sliding along the master surface to snag on these corners. Therefore, Abaqus/Standard automatically smooths the master surface for contact calculations utilizing node-to-surface discretization to minimize this phenomenon. The details are discussed further in "Smoothing master 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 master surface penetrating the slave 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," Section 36.1.3).

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," Section 33.6.1) 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," Section 36.1.2.

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 "Defining a precise initial clearance or overclosure for small-sliding contact" in "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5).
- Various usages of special-purpose contact elements, such as tube-to-tube contact elements (see "Contact modeling with elements," Section 38.1.1, and "Tube-to-tube contact elements," Section 38.3.1), 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 36.1.1–5, with surface **ASURF** acting as the slave surface to surface **BSURF** in a finite-sliding, node-to-surface contact pair.

In this example slave node 101 may come into contact anywhere along the master 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 master surface **BSURF** as the bodies deform. Figure 36.1.1–6 shows the possible evolution of the contact between node 101 and its master 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.

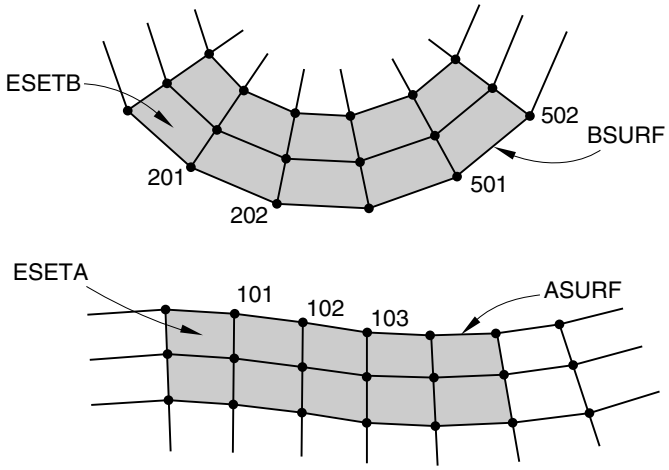


Figure 36.1.1–5 Contacting bodies.

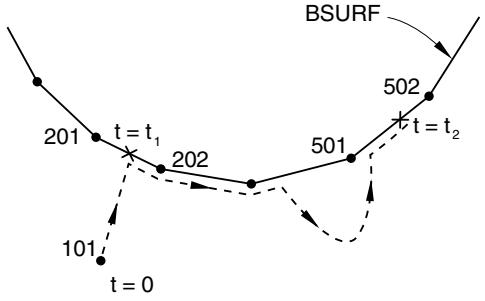


Figure 36.1.1–6 Trajectory of node 101 in finite-sliding contact.

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 slave surface with respect to the master surface within each increment and allows for double-sided shell and membrane master surfaces. The path-based tracking algorithm is available only for finite-sliding, surface-to-surface contact interactions involving element-based master 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 master surface or if the incremental motion cuts across corners of the master 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 master surface;
- a non-default option for finite-sliding, surface-to-surface contact pairs involving an element-based master 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 master surfaces for the finite-sliding, node-to-surface formulation

The finite-sliding, node-to-surface contact formulation requires that master surfaces have continuous surface normals at all points. Convergence problems can result if master surfaces that do not have continuous surface normals are used in finite-sliding, node-to-surface contact analyses; slave nodes tend to get "stuck" at points where the master surface normals are discontinuous. Abaqus/Standard automatically smooths the surface normals of element-based master surfaces (see "Smoothing deformable master 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," Section 2.3.4). No such smoothing of master surface normals is needed with the finite-sliding, surface-to-surface formulation.

Smoothing deformable master surfaces and rigid surfaces defined with rigid elements

For finite-sliding, node-to-surface contact simulations with planar or axisymmetric deformable master 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 36.1.1–7 for first-order elements (linear segments) and in Figure 36.1.1–8 for second-order elements (parabolic segments). For finite-sliding, node-to-surface simulations with three-dimensional deformable master surfaces and rigid master surfaces using rigid elements, Abaqus/Standard will smooth any discontinuous surface normal transitions between the master surface facets.

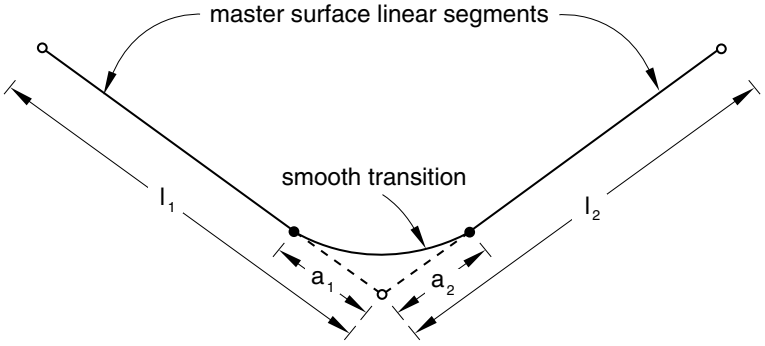


Figure 36.1.1–7 Smoothing between linear segments.

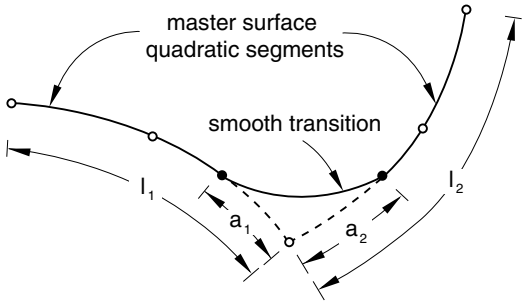


Figure 36.1.1–8 Smoothing between quadratic segments.

You can control the degree of smoothing of the master 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 master 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 36.1.1–7 and Figure 36.1.1–8). 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 master 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 master surfaces, f is defined as a fraction of the dimension of a facet as shown in Figure 36.1.1–9. 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 36.1.1–7 and Figure 36.1.1–8. 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," Section 29.3.1) 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.

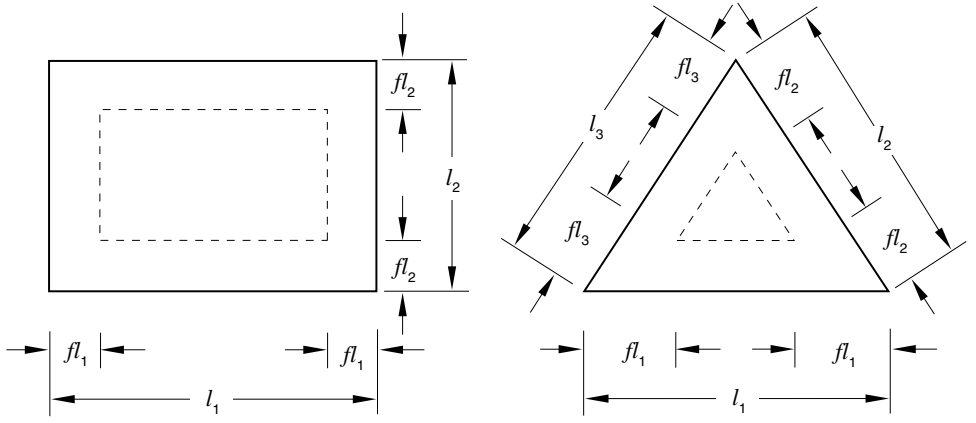


Figure 36.1.1–9 Smoothing of a three-dimensional master surface.

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 master surface:** f

Smoothing a deformable master surface along symmetry edges

When a two-dimensional or axisymmetric deformable master 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 master surface regardless of whether a symmetry boundary condition is defined. The small- and finite-sliding, surface-to-surface formulations have no special treatment for master surfaces ending at a symmetry plane. See "Modifying the master surface normals" in "Contact formulations in Abaqus/Standard," Section 36.1.1, for a discussion of how the small-sliding, node-to-surface formulation treats master surfaces ending at a symmetry plane.

Overriding the default smoothing behavior for finite-sliding, node-to-surface contact

To model a master 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 slave node is in contact with an interior corner or fold in the master surface.

To accurately model the master surface with a corner shown in Figure 36.1.1–10, you must define two contact pairs: the first contact pair has **ASURF** as the slave surface and **BSURFA** as the master surface; the second contact pair has **ASURF** as the slave surface and **BSURFB** as the master surface.

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" in "General and linear perturbation procedures," Section 6.1.2). 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 master surfaces with large differences in surface normals between facets.

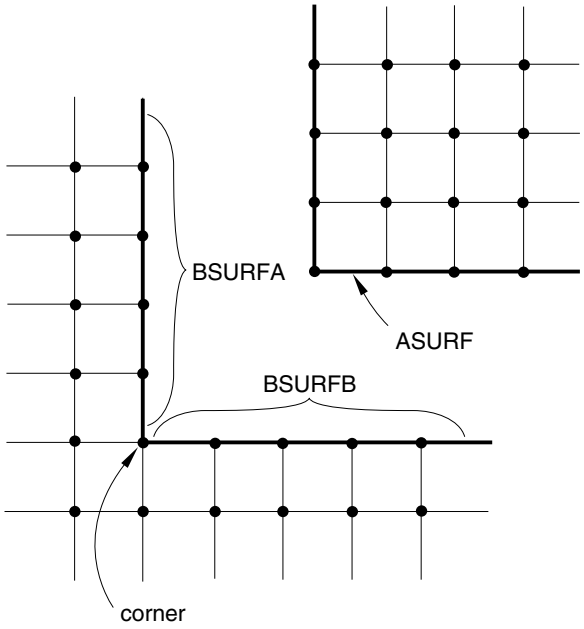


Figure 36.1.1–10 Master surface with a corner.

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 master and slave 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" in "Procedures: overview," Section 6.1.1).

Contact simulations that involve strong frictional effects can also produce unsymmetric terms. See "Unsymmetric terms in the system of equations" in "Frictional behavior," Section 35.1.5, 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 slave node will interact with the same local area of the master 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 master surface with each slave node of a small-sliding contact pair. Contact interactions are considered between a given slave node (or

region nearby a given slave 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 36.1.1–11 (for example, the slave 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 master surface and an orientation vector at the anchor point (see Figure 36.1.1–11).

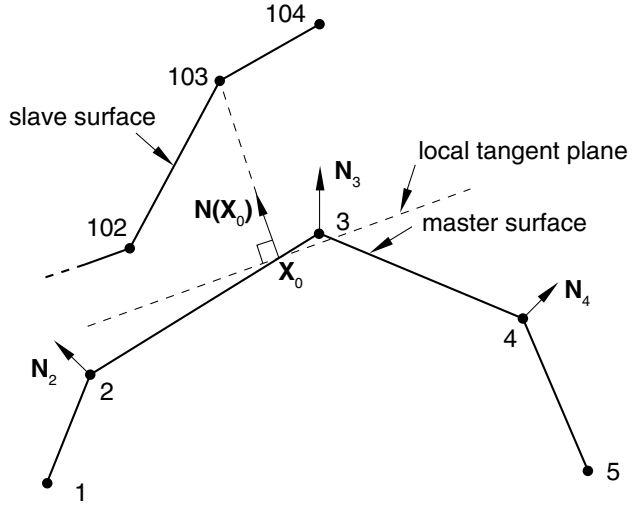


Figure 36.1.1–11 Definition of the anchor point and local tangent plane used by the small-sliding, node-to-surface formulation for node 103.

The algorithm used to define anchor points is described below. If an anchor point cannot be determined for a particular slave node, no contact constraint will be enforced for that slave node.

Having a local tangent plane for each slave node means that for the small-sliding tracking approach Abaqus/Standard does not have to monitor slave nodes for possible contact along the entire master 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 slave node's local tangent plane such that the vector from the anchor point to the slave node coincides with a smoothly varying normal vector on the master surface. The anchor point is chosen before the analysis starts using the initial configuration of the model.

Smoothly varying master surface normals

The algorithm requires that the master surface have a smoothly varying normal vector $\mathbf{N}(\mathbf{x})$, where $\mathbf{x}$ is any point on the master surface. The first step in defining $\mathbf{N}(\mathbf{x})$ is to construct the unit normal vectors at

each node of the master surface. Abaqus/Standard forms these nodal normals by averaging the normals of the element faces making up the master 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 36.1.1–11 shows the nodal unit normals for a master surface, the anchor point $\mathbf{X}_0$, and the local tangent plane associated with slave 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 slave 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 master surface normals

Sometimes the default smoothed master surface normal and the local tangent plane that Abaqus/Standard calculates are not suitable for the desired analysis. The most common situation where unsuitable surface normals are calculated occurs when a curved master 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 Abaqus/Standard and Abaqus/Explicit," Section 32.3.1). 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 slave node does not have a normal from the master surface pass through it (the slave node is said not to "intersect" the master surface). No contact constraints will be enforced for such slave nodes.

A message is printed in the data (**`.dat`**) file whenever a slave node does not intersect its master surface. By specifying the proper symmetry "type" boundary condition, Abaqus/Standard will calculate the correct normal and local tangent planes along the symmetry planes of the master surface.

If the smoothed normals of the master surface and the local tangent planes calculated by Abaqus/Standard are unsuitable and it is not feasible to apply symmetry "type" boundary conditions, several other methods are available for modifying the smoothed normals. One method is to add or remove some of the element faces making up the master surface. However, this method can influence only the surface normals near the perimeter of the master surface.

The other method is to modify the nodal normals on the master surface by defining user-specified normals (see "Normal definitions at nodes," Section 2.1.4). This method is especially useful in providing a more accurate representation of the surface geometry. Figure 36.1.1–12 shows two concentric cylinders that contact each other; the inner cylinder is chosen as the master surface **`CSURF`**.

If a half-symmetry model is used, the default master surface normal at the symmetry plane will cause problems. As shown in Figure 36.1.1–12, the nodal normal $\mathbf{N_1}$ does not point along the symmetry plane, which means that slave node 100 will never intersect the master surface. In a small-sliding problem if a slave node fails to intersect the master surface at the start of the analysis, it will be free to penetrate the master surface because no local tangent plane will be formed. Abaqus/Standard provides the initial contact status—open, overclosed, or "no intersection"—in the data file for every slave node in the model

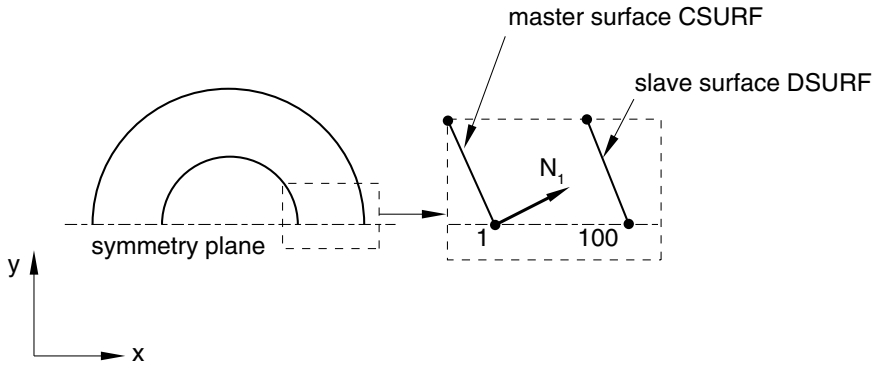


Figure 36.1.1–12 Master surface normal at node 1 in a small-sliding model of concentric cylinders. With the default $\mathbf{N_1}$ slave node 100 will never contact **CSURF**.

(see "Contact diagnostics in an Abaqus/Standard analysis," Section 37.1.1). Use this information to confirm that the necessary tangent planes for a model have been found.

In situations such as that shown in Figure 36.1.1–12, define a YSYMM "type" boundary condition at node 1 to specify the symmetry plane. The master normal at the node on the symmetry plane will be modified to lie along the symmetry plane, allowing slave node 100 to see the master surface **CSURF**. The small- and finite-sliding, surface-to-surface formulations have no special treatment for master surfaces ending at a symmetry plane. See "Smoothing a deformable master surface along symmetry edges" in "Contact formulations in Abaqus/Standard," Section 36.1.1, for a discussion of how the finite-sliding, node-to-surface formulation treats master surfaces ending at a symmetry plane.

In situations where a symmetry "type" boundary condition cannot be specified, define a user-specified normal (1.00E+00, 0.00E+00, 0.00E+00) at node 1 on the master surface **CSURF** to correct the problem. This method will also allow slave node 100 to see the master surface.

The modification to **CSURF**'s normal at node 1, which makes **CSURF** a better approximation of the actual surface, is shown in Figure 36.1.1–13.

Small-sliding, surface-to-surface contact

A key difference with the surface-to-surface approach is that more than one slave node is involved in each contact constraint—this is related to the fact that the surface-to-surface formulation enforces contact conditions in an average sense over regions nearby slave nodes rather than only at individual slave 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 master surface per averaging region of the slave surface. The constraint participation factors for the slave nodes remain constant for small-sliding contact. The effective center-of-action on the slave surface per contact constraint may differ slightly from the location of the predominant slave node associated with the constraint.

The small-sliding, surface-to-surface contact formulation determines master anchor points and normal directions in a manner similar to that used by the small-sliding, node-to-surface contact

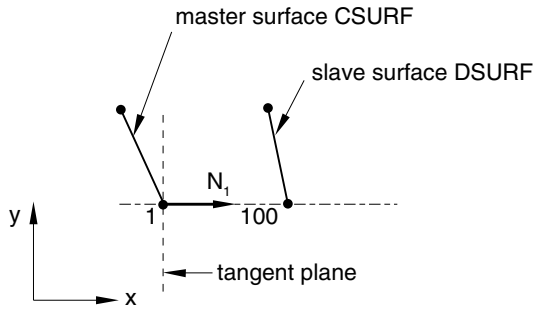


Figure 36.1.1–13 The modified master surface normal at node 1 of **CSURF** now allows slave node 100 to contact **CSURF**.

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 master surface where the averaging region of the slave projects onto the master surface. This projection occurs along the slave surface normal direction. This method does not make use of smoothed master 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 slave surface is a node-based surface;
- if the projection along the slave surface normal direction does not intersect the master surface (but an anchor point can be found using the interpolated master surface normal direction algorithm discussed above for the small-sliding, node-to-surface formulation); or
- if single-sided slave and master 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 slave node, and the tangent plane is normal to this direction.

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" in "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5).

Once the contact direction is defined, the orientation of the local tangent plane with respect to the master 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 master surface is deformable, the deformation of the master surface. The position of the anchor point relative to the surrounding nodes on the master surface facet does not change as the master surface deforms.

Load transfer

In a small-sliding analysis each constraint can transfer load only to a limited number of nodes on the master surface. These nodes on the master surface are chosen based on their initial proximity to the anchor point. The magnitude of load transferred to each master surface node is based on proximity in the current, deformed configuration to the center-of-action on the slave surface (which corresponds to a slave node for the node-to-surface formulation). For example, in Figure 36.1.1–11 node 103 transmits load to both nodes 2 and 3 on the master surface if node-to-surface discretization is used (if surface-to-surface discretization is used, load may be transmitted to additional nearby master nodes). Thus, if node 103 contacts the local tangent plane, a larger share of the force would be transmitted to the master surface node, 2 or 3, closer to the slave node.

When the anchor point $\mathbf{X}_0$ corresponds to a node on the master surface, as is the case with slave node 104 and master surface node 3 in Figure 36.1.1–11, the transmitted load for node-to-surface contact is shared by the node at $\mathbf{X}_0$ and all of the master surface nodes that share an adjacent surface facet with that node (additional master nodes may take part in the load transfer for surface-to-surface contact). In Figure 36.1.1–11 the three master surface nodes sharing the force transmitted by slave node 104 are nodes 2, 3, and 4.

As the center-of-action on the slave surface for a constraint slides along its local tangent plane, Abaqus/Standard updates the distribution among the master surface nodes. However, no additional master 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 master surface nodes, regardless of the sliding distance. Figure 36.1.1–14 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 slave node 101 in Figure 36.1.1–5 and its master surface **BSURF**. Using the unit normal vectors $\mathbf{N}_{201}$ and $\mathbf{N}_{202}$, the anchor point $\mathbf{X}_0$ is found for slave 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 36.1.1–14, it will continue to transmit load to nodes 201 and 202 when, in fact, it really slid off the mesh forming the master 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:

- Slave nodes should slide less than an element length from their corresponding anchor point and still be contacting their local tangent plane. If the master surface is highly curved, the slave nodes should slide only a fraction of an element length. The accumulated slip at a slave node (CSLIP) can provide a good estimate of how far a slave node has moved.

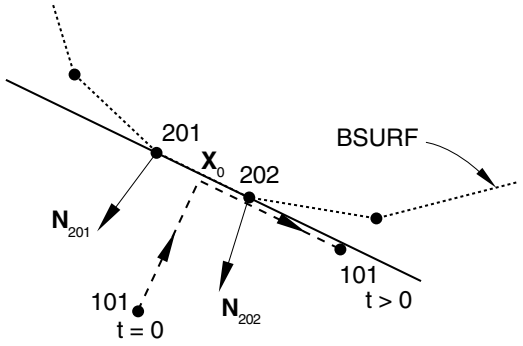


Figure 36.1.1–14 Excessive sliding in a small-sliding contact analysis.

- 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," Section 2.1.4) to improve the smoothly varying master surface normal, $\mathbf{N}(\mathbf{x})$.
- The rotation and deformation of the master surface should not cause the local tangent planes to become a poor representation of the master surface during the course of the analysis.

Choosing the master and slave surfaces in small-sliding problems

The basic guidelines given in "Defining contact pairs in Abaqus/Standard," Section 34.3.1, should still be followed in a small-sliding simulation—the slave 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 master surface. With small-sliding contact each slave node views the master 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 master surface in the initial configuration, but deformation and rotation of the master surface can reorient the local tangent planes such that they become a poor representation of the master surface. Figure 36.1.1–15 shows an example where distortion of the master surface results in such a situation. This problem can be minimized to some extent by using a more refined mesh on the master 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 master surface node remain constant during an infinitesimal-sliding simulation.

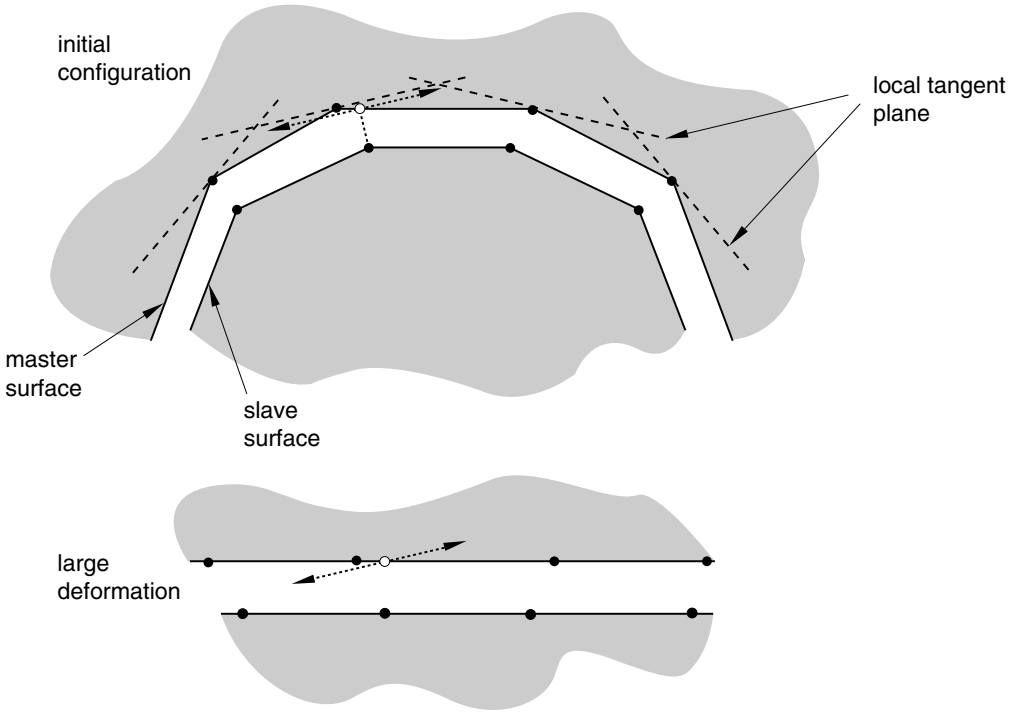


Figure 36.1.1–15 Master surface deformation in a small-sliding contact analysis can cause problems with the local tangent planes.

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 (sometimes called "slip directions") 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 master 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" in "Choosing the element's dimensionality," Section 26.1.2.

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 slave surface normal, using the standard convention for calculating surface tangents (see "Conventions," Section 1.2.2) with the assumption that the contact normal corresponds to the negative normal to the slave surface.
- **Finite-sliding, node-to-surface formulation:** For contact involving a slave 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 slave 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 slave surfaces the tentative directions are based on the slave surface using the standard convention for calculating surface tangents. For node-based slave 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 master surface normal at the tracked point on the master surface.
- **Small-sliding, surface-to-surface formulation:** The default initial orientations of the two local tangent directions are based on the slave surface normal, using the standard convention for calculating surface tangents, except for contact involving analytical rigid surfaces, in which case the local tangent directions are based on the master surface normal.
- **Small-sliding, node-to-surface formulation:** The default initial orientations of the two local tangent directions are calculated at each point on the master surface based on the master 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," Section 2.2.5) 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 slave 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 slave surface and an analytical rigid surface.

An orientation that is defined on a slave 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 slave surface undergoes finite motion. In this case a warning message is issued during input processing.

Input File Usage: ∗CONTACT PAIR, INTERACTION=*interaction_property_name*
slave surface name, *master surface name*, *orientation for slave surface*
slave surface name, *master surface name*, , *orientation for master 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 master 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 master surface when the slave node comes into contact.

36.1.2 CONTACT CONSTRAINT ENFORCEMENT METHODS IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Defining general contact interactions in Abaqus/Standard," Section 34.2.1
- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- "Mechanical contact properties: overview," Section 35.1.1
- "Contact pressure-overclosure relationships," Section 35.1.2
- *SURFACE BEHAVIOR
- *CONTACT CONTROLS
- "Defining general contact," Section 15.13.1 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a contact interaction property," Section 15.14.1 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

Contact constraint enforcement methods in Abaqus/Standard:

- are specified as part of the surface interaction definition;
- determine how contact constraints imposed by a physical pressure-overclosure relationship (see "Contact pressure-overclosure relationships," Section 35.1.2) 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," Section 35.1.5. 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 (including general contact) if a "hard" pressure-overclosure relationship is in effect.
- 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," Section 35.1.2).
- 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," Section 33.6.1) should be avoided for directly enforced hard contact.

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:

*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)**

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," Section 35.1.2.

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 in the following situations:

- finite-sliding, surface-to-surface formulations if constraint positions at faces is specified (non-default); and
- three-dimensional self-contact using node-to-surface discretization.

In both of these instances you can use an alternate enforcement method or the direct method with a softened pressure-overclosure relationship.

You may experience similar overconstraint problems with symmetric master-slave contact pairs (see "Using symmetric master-slave contact pairs to improve contact modeling" in "Defining contact pairs in Abaqus/Standard," Section 34.3.1). 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" in "Common difficulties associated with contact modeling in Abaqus/Standard," Section 37.1.2, 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 36.1.2–1.

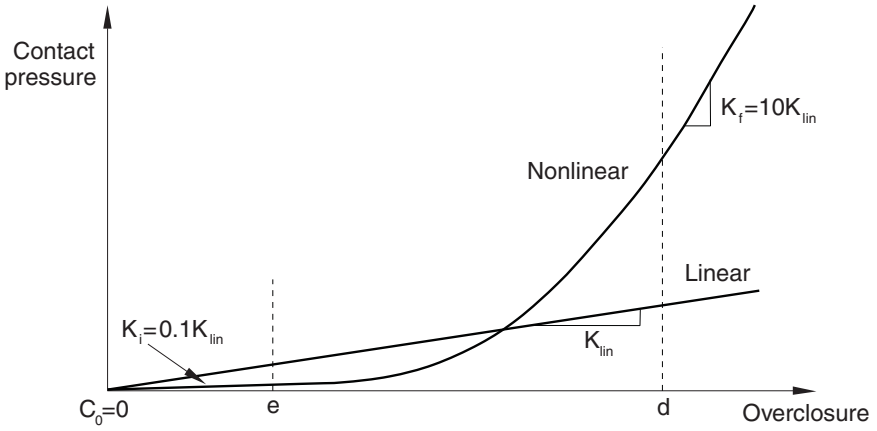


Figure 36.1.2–1 Comparison of linear and nonlinear pressure-overclosure relationships with default settings.

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.

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**

Nonlinear penalty method

With the nonlinear penalty method, the pressure-overclosure curve has four distinct regions shown in Figure 36.1.2–2.

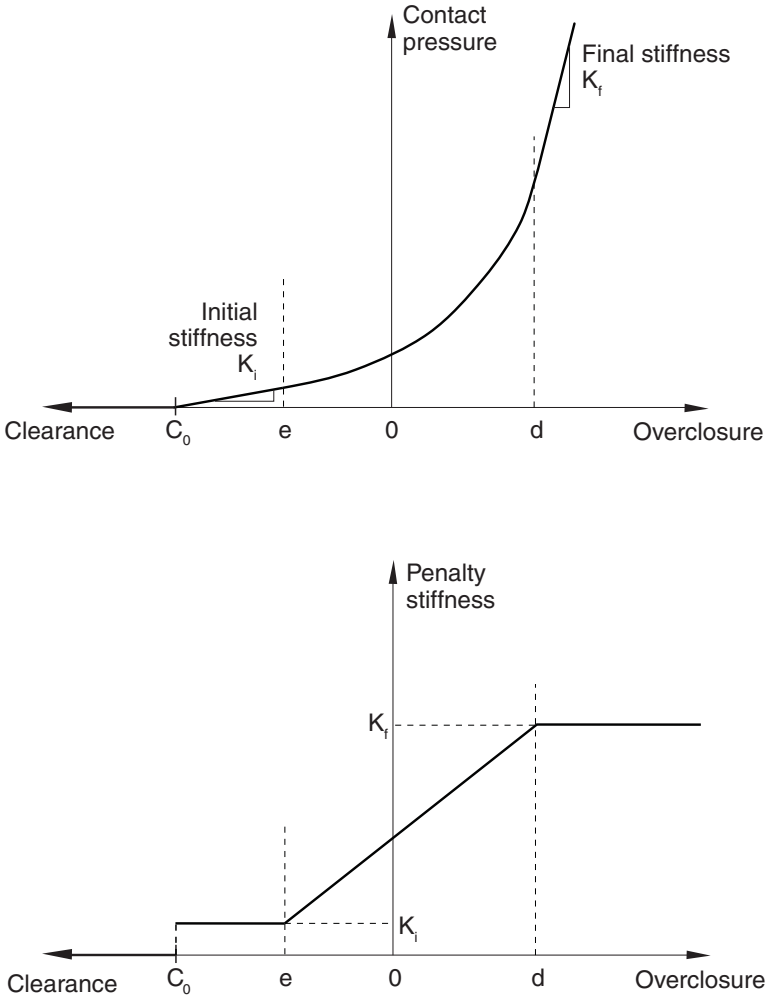


Figure 36.1.2–2 Nonlinear penalty pressure-overclosure relationship.

- 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.

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**

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 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*

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 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*

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*

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 “Design sensitivity analysis,” Section 18.1.1)
- Direct steady-state dynamic analysis (see “Direct-solution steady-state dynamic analysis,” Section 6.3.4)
- Quasi-Newton method (see “Convergence criteria for nonlinear problems,” Section 7.2.3)

If surface elements have been used to define a contact surface on the exterior of a substructure (see “Contact modeling if substructures are present,” Section 34.3.9), 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 slave node penetrates the master 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 may 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:

*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)**

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 may 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 may result in an excessive number of augmentation iterations.

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*

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 "Design sensitivity analysis," Section 18.1.1)
- Direct steady-state dynamic analysis (see "Direct-solution steady-state dynamic analysis," Section 6.3.4)
- Quasi-Newton method (see "Convergence criteria for nonlinear problems," Section 7.2.3)

If surface elements have been used to define a contact surface on the exterior of a substructure (see "Contact modeling if substructures are present," Section 34.3.9), 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.

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 36.1.2–1 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 may change as the contact status changes.

Table 36.1.2–1 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 $k_{penalty} \leq 1000k_e$

Constraint Method	Use Lagrange Multipliers	
	Yes	**No**[1]
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.

36.1.3 SMOOTHING CONTACT SURFACES IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Defining general contact interactions in Abaqus/Standard," Section 34.2.1
- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- *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. 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 Abaqus/Standard 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. Furthermore, discontinuities in the surface normal direction at surface facet boundaries can contribute to convergence difficulties.

Abaqus/Standard offers techniques for overcoming the accuracy and convergence difficulties associated with faceted surfaces in contact interactions. These techniques allow a discretized surface with discontinuous surface normals to more closely approximate the behavior of a smooth surface with continuous normals during an analysis. The smoothing technique used in node-to-surface contact is different from the smoothing technique used in surface-to-surface and general contact:

- Node-to-surface contact smoothing is applied by default and affects the entire master surface.
- Surface-to-surface contact smoothing is not applied by default, but it can be applied to any surface regions whose geometry is roughly axisymmetric.

Surface-to-surface contact typically gives the most accurate results.

Smoothing master surfaces for node-to-surface contact pairs

Surface smoothing in node-to-surface contact pairs improves numerical stability and sometimes improves solution accuracy. Slave nodes traveling along a master surface tend to "snag" on sharp corners, resulting in convergence difficulties. Because of this behavior, Abaqus/Standard automatically smooths the master surface in node-to-surface contact pairs. This smoothing technique recalculates the master surface normals along facet edges and, depending on the type of surface, may affect the

surface geometry. The details of smoothing for node-to-surface contact formulations are discussed in "Smoothing master surfaces for the finite-sliding, node-to-surface formulation" in "Contact formulations in Abaqus/Standard," Section 36.1.1, and "Using the small-sliding tracking approach" in "Contact formulations in Abaqus/Standard," Section 36.1.1.

Smoothing contact surfaces for surface-to-surface contact

Smooth surfaces are not usually necessary in surface-to-surface contact to ensure analysis convergence; therefore, no smoothing is applied to these surfaces by default. However, an optional smoothing technique is available for improving the contact stress and pressure accuracy for axisymmetric (or nearly axisymmetric) surfaces in surface-to-surface contact interactions.

Surface-to-surface contact smoothing can be applied to specific surface regions. These regions must be roughly axisymmetric (all points on the surface are nearly equidistant from a single axis) or roughly spherical (all points on the surface are nearly equidistant from a single point). The pin insertion model in Figure 36.1.3–1 could benefit from surface-to-surface contact smoothing: the body of the pin and the hole are axisymmetric surfaces, and the head of the pin is a spherical surface. Surface-to-surface contact smoothing would also be effective if the surfaces were not perfectly axisymmetric or spherical; for example, if the pin body were slightly elliptical.

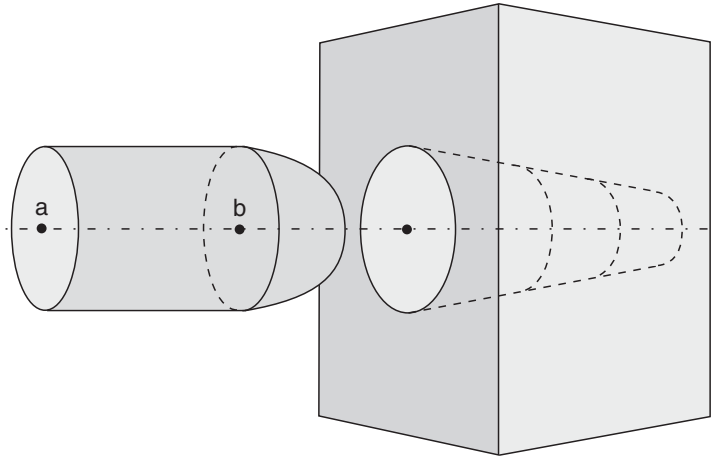


Figure 36.1.3–1 Surface-to-surface contact model with surface smoothing.

Applying contact smoothing to surface-to-surface contact pairs

Surface-to-surface contact smoothing for contact pairs 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. Two geometry correction methods can be employed:

- A 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.
- A spherical smoothing method is applicable to surfaces approximating a portion of a sphere in three dimensions.

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 master surfaces and to slave 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 smoothing) or the approximate spherical center (for spherical smoothing).

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):
slave_region, *master_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):
slave_region, *master_region*, SPHERICAL, X_a, Y_a, Z_a

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 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 geometry surfaces**

Surface-to-surface contact smoothing cannot be applied to surfaces on orphan mesh models in Abaqus/CAE.

Example

To improve contact pressure accuracy for the model in Figure 36.1.3–1, contact smoothing can be applied to both the master and slave surfaces. Two different geometric correction methods are required for the pin (the slave surface), so additional surfaces are defined corresponding to regions of the slave 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, a single circumferential smoothing technique 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, Xb, Yb, Zb
PIN_BODY, HOLE, CIRCUMFERENTIAL, Xa, Ya, Za, Xb, Yb, Zb
```

Applying contact smoothing to general 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. General contact uses the same geometry correction methods as contact pairs:

- A 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.
- A spherical smoothing method is applicable to surfaces approximating a portion of a sphere in three dimensions.

Input File Usage: ∗SURFACE PROPERTY ASSIGNMENT, PROPERTY=GEOMETRIC CORRECTION
data lines to define smoothing regions (see below)

Use the following data line 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

Use the following data line 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

Repeat the data lines as many times as necessary to define the appropriate geometry corrections for all surfaces in the contact domain.

Abaqus/CAE Usage: Contact surface smoothing can be applied only to native geometry models in Abaqus/CAE. By default, Abaqus/CAE automatically detects all 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, or select **NONE** to prevent smoothing of the surface.

Considerations for using surface-to-surface contact smoothing

The surface-to-surface contact smoothing technique assumes that the initial locations of surface nodes lie on the true initial surface geometry, with the exception of midside nodes of higher-order elements. This smoothing technique remains 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).

The effects of surface-to-surface contact smoothing 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 this smoothing technique typically has 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 surface-to-surface contact smoothing for large-deformation analyses. The effectiveness of surface-to-surface contact 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.

Effects of contact surface smoothing

The impact of contact surface smoothing can be demonstrated by a simple model of an interference fit between concentric cylinders modeled with first-order elements of different sizes, as shown in Figure 36.1.3–2. Discrepancies between the true surface geometry and the faceted surface geometry result in noise in the contact pressure solution. 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.

Modeling the interference fit with a surface-to-surface contact pair and using circumferential contact smoothing consistently yields low-noise pressure results that are within 3% of the analytical

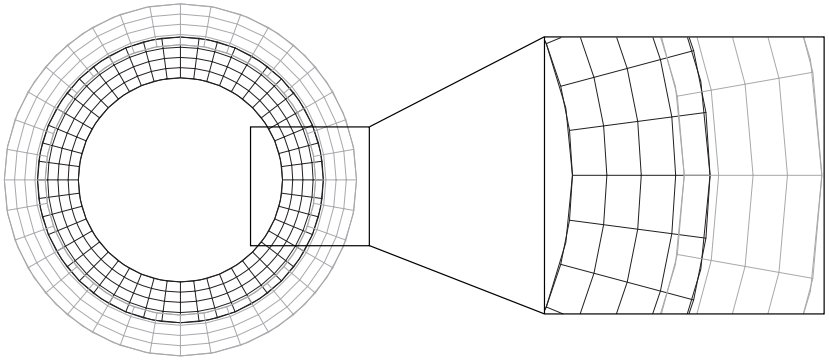


Figure 36.1.3–2 Initial mesh geometry for interference fit model.

solution, regardless of the size of the interference distance. The effect is drastically noticeable for small-deformation analyses, but improvements can be observed even for larger deformations.

For a node-to-surface contact pair, increasing the smoothing fraction to the maximum value of 0.5 marginally reduces the noise in the pressure solution in a two-dimensional model. Increasing the smoothing factor in a three-dimensional model has little effect on accuracy, since physical surfaces are not smoothed for three-dimensional node-to-surface smoothing; see "Smoothing master surfaces for the finite-sliding, node-to-surface formulation" in "Contact formulations in Abaqus/Standard," Section 36.1.1, for more information.

36.2 Contact formulations and numerical methods in Abaqus/Explicit

- "Contact formulation for general contact in Abaqus/Explicit," Section 36.2.1
- "Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2
- "Contact constraint enforcement methods in Abaqus/Explicit," Section 36.2.3

36.2.1 CONTACT FORMULATION FOR GENERAL CONTACT IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Defining general contact interactions in Abaqus/Explicit," Section 34.4.1
- *CONTACT
- *CONTACT FORMULATION
- "Specifying master-slave assignments for general contact," Section 15.13.5 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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," Section 36.2.3.

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 master-slave contact the node with the constraint is part of the slave surface and the surface with which it interacts is called the master surface. For balanced master-slave 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 master surface and once with the second surface acting as the master surface. The weighted average of the two corrections (or forces) is applied to the contact interaction.

Balanced master-slave 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 master-slave contact the nodes on the master surface can, in principle, penetrate the slave surface unhindered (see Figure 36.2.1–1).

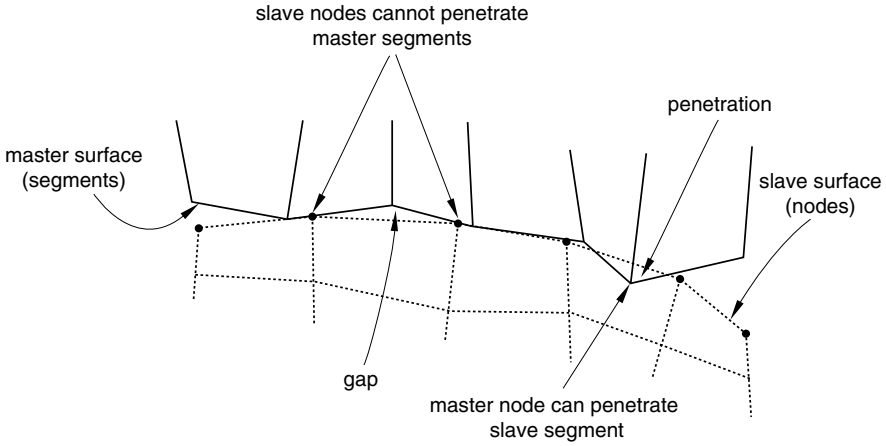


Figure 36.2.1–1 Master surface penetrations into the slave surface in pure master-slave contact due to coarse discretization.

The general contact algorithm in Abaqus/Explicit uses balanced master-slave weighting whenever possible; pure master-slave weighting is used for contact interactions involving node-based surfaces, which can act only as pure slave surfaces and for contact interactions involving analytical rigid surfaces, which can act only as pure master surfaces. Surface-based cohesive behavior also always uses a pure master-slave algorithm. However, you can choose to specify a pure master-slave weighting for other interactions as well.

There is no master-slave relationship for edge-to-edge contact; both contacting edges are given equal weighting.

Specifying pure master-slave weighting for node-to-face contact

You can specify that a general contact interaction should use pure master-slave 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 master surface. When two originally flat surfaces contact one another, a more uniform penetration distance distribution (and consequently pressure distribution) may result with pure master-slave weighting where the more refined surface acts as the slave surface as compared to balanced master-slave 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 will automatically generate contact exclusions for the master-slave orientation opposite to that specified; therefore, node-to-face self-contact will be 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 master-slave weighting with *surf_A* considered to be the slave surface would result in exclusions being generated internally for faces of *surf_A* contacting nodes of *surf_B*; node-to-face self-contact would be excluded for the region of overlap between *surf_A* and *surf_B*. A warning message will be 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 node–face self-contact for the surface.

Input File Usage: Use the following option to indicate that the first surface should be considered the slave surface (default):

```
*CONTACT FORMULATION, TYPE=PURE MASTER-SLAVE
surf_1, surf_2, SLAVE
```

Use the following option to indicate that the first surface should be considered the master surface:

```
*CONTACT FORMULATION, TYPE=PURE MASTER-SLAVE
surf_1, surf_2, MASTER
```

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 master-slave 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 master-slave assignments.

In the **First Surface Type** column, enter SLAVE to indicate that the first surface should be considered the slave surface, and enter MASTER to indicate that the first surface should be considered the master surface.

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 may be gained by converting double-sided elements to single-sided if there is a chance that slave nodes may 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 may 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.

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,” Section 36.2.2).

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 master 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 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.

36.2.2 CONTACT FORMULATIONS FOR CONTACT PAIRS IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Surfaces: overview," Section 2.3.1
- "Defining contact pairs in Abaqus/Explicit," Section 34.5.1
- *CONTACT PAIR
- "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

Overview

The contact formulation for the contact pair algorithm in Abaqus/Explicit includes:

- the contact surface weighting (balanced or pure master-slave); 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," Section 36.2.3.

Contact surface weighting

Both the pure master-slave and the balanced master-slave 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 master-slave, kinematic contact algorithm, by default, in the following situations (the first surface in each situation listed is designated the master 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 master-slave, 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 master-slave weighting, by default, in the following situations (the first surface in each situation listed is designated the master 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 master-slave 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 master-slave 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 master-slave 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 slave surface. Contact-induced noise can occur if a surface on a much denser body is at all weighted as a slave surface. Finally, to avoid significant penetration for hard contact, the surface with the finer mesh should not be the master surface in the pure master-slave contact pair.

When the penalty contact method is chosen, you can choose to specify a pure master-slave 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 master-slave weighting as compared to balanced master-slave 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 master-slave weighting provides better enforcement of contact constraints in most cases.

You define a weighting factor, f, to specify the master-slave weighting. Set f=1.0 to designate the first surface in the contact pair as the master surface and the second surface as the slave surface. Set f=0.0 to designate the first surface in the contact pair as the slave surface and the second surface as the master surface. Specifying any value of f between 0 and 1.0 invokes the balanced master-slave contact algorithm. When f=0.5, which is the default for balanced master-slave contact pairs, Abaqus/Explicit weights each set of corrections equally. In contrast, Abaqus/Standard uses a pure master-slave contact algorithm; the slave 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 36.2.2–1, with **ASURF** acting as the slave surface:

```
*SURFACE,NAME=ASURF
ESETA,
*SURFACE,NAME=BSURF
ESETB,
*CONTACT PAIR,INTERACTION=PAIR1, WEIGHT=0.0
ASURF, BSURF
*SURFACE INTERACTION,NAME=PAIR1
```

In the example shown in Figure 36.2.2–1 slave node 101 may come into contact anywhere along the master 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 master surface **BSURF** as the bodies deform. Figure 36.2.2–2 shows the possible evolution of the contact between node 101 and its master 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.

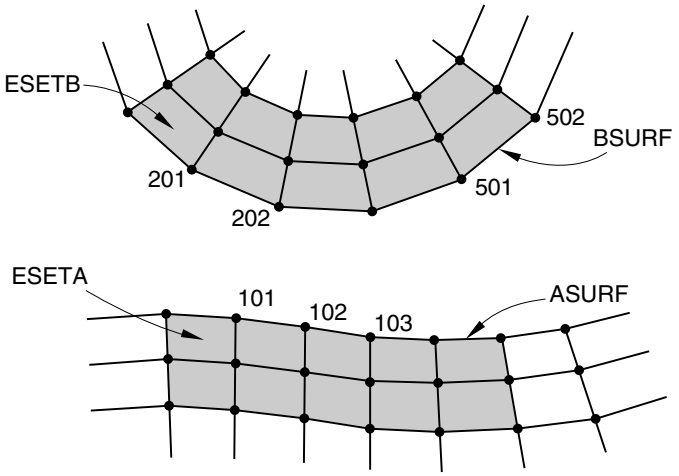


Figure 36.2.2–1 Contacting bodies.

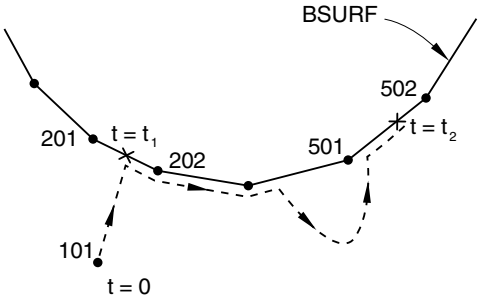


Figure 36.2.2–2 Trajectory of node 101 in finite-sliding contact.

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" in "General and linear perturbation procedures," Section 6.1.2). 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 slave node will interact with the same local area of the master 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 slave node interacts with its own local tangent plane on the master surface (see Figure 36.2.2–3). The slave 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 master surface and an orientation vector at the anchor point (see Figure 36.2.2–3).

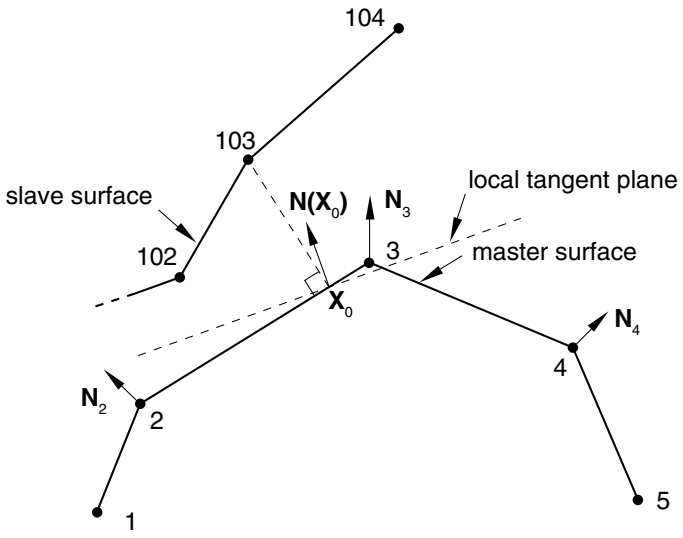


Figure 36.2.2–3 Definition of the anchor point and local tangent plane for node 103.

Having a local tangent plane for each slave node means that for the small-sliding formulation Abaqus/Explicit does not have to monitor slave nodes for possible contact along the entire master 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 master-slave 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 36.2.2–1:

```
*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: **NIgeom: 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 master surface facet for all steps in which the contact pair is active. No contact constraints are enforced for slave nodes whose nearest point lies on the free perimeter of the master surface in the original configuration and that do not project onto any master surface facet.

Abaqus/Explicit chooses the anchor point as the nearest point on the master surface. The orientation of the tangent plane is calculated by default from the normals at the master surface nodes, or you can specify it directly.

- Master surface normals: The first step in defining the tangent plane orientation is to construct the unit normal vectors at each node of the master surface. Abaqus/Explicit forms these nodal normals by averaging the normals of the element faces making up the master surface; only the element faces in the surface definition will contribute to the nodal normals. The tangent plane orientation is then calculated from the master surface nodal normals and the element shape functions at the anchor point.

  Figure 36.2.2–3 shows the nodal unit normals for a master surface, the anchor point $\mathbf{X}_0$, and the local tangent plane associated with slave node 103. Abaqus/Explicit uses the closest point on the master surface as the anchor point. $\mathbf{N}(\mathbf{X_0})$ is the contact direction for slave 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.
- Master surface normals at symmetry planes: Sometimes the master 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 master 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 Abaqus/Standard and Abaqus/Explicit," Section 32.3.1). 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 slave node does not project onto any master surface facet (the slave node is said not to "intersect" the master surface). No contact constraints will be enforced for such slave 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 master surfaces ending at a symmetry plane.

Figure 36.2.2–4 shows two concentric cylinders that contact each other; the inner cylinder is chosen as the master 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 slave node 100 has no anchor point within the perimeter of the master 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, slave node 100 will be free to penetrate the master surface. If the symmetry "type" format is used, the master 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 36.2.2–5. Defining a YSYMM "type" boundary condition at node 1 to specify the symmetry plane will allow slave node 100 to see the master surface **CSURF**.

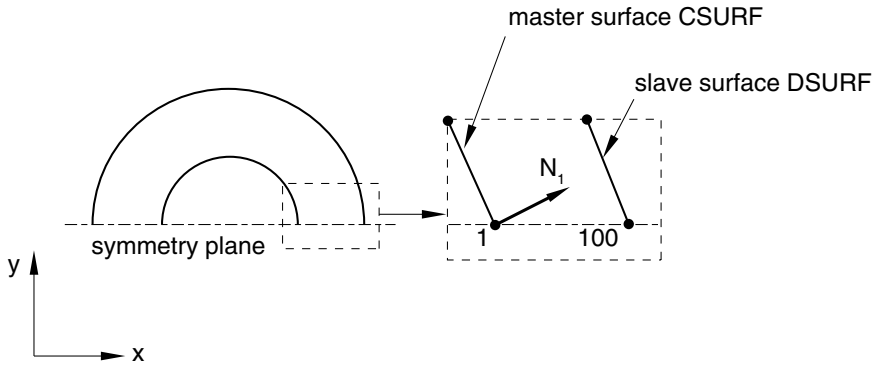


Figure 36.2.2–4 Master surface normal at node 1 in a small-sliding model of concentric cylinders. With the default $\mathbf{N_1}$ slave node 100 will never contact **CSURF**.

- Modifying the local tangent plane orientation: In some cases the contact direction, $\mathbf{N}(\mathbf{X_0})$, defined from the master 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 36.2.2–6 shows how the averaged master normals will not be oriented correctly in the radial direction. In this case you should specify the contact direction directly for each slave node by defining spatially varying initial clearances (see "Specifying initial clearance values precisely" in "Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4). The location of the anchor point is not affected by reorienting the tangent plane using an initial clearance definition.

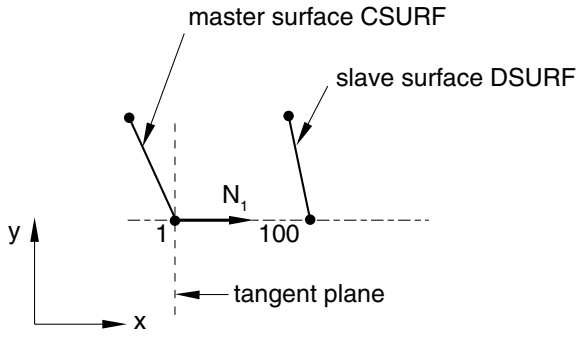


Figure 36.2.2–5 The modified master surface normal at node 1 of **CSURF** now allows slave node 100 to contact **CSURF**.

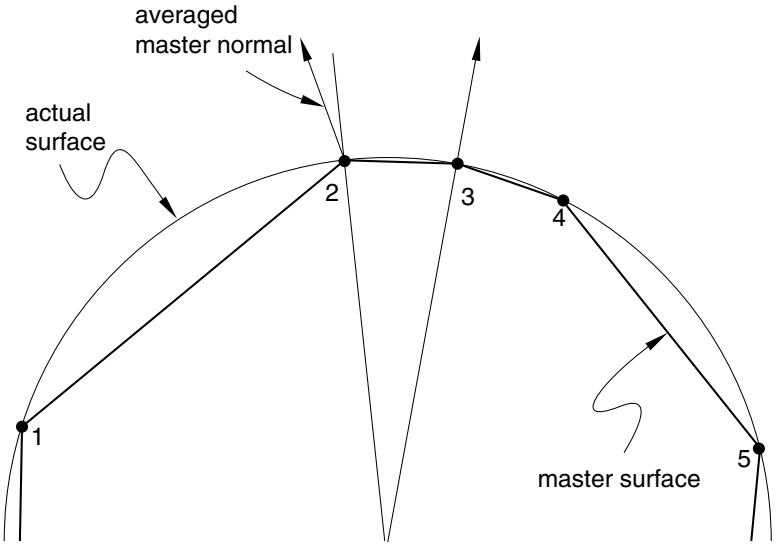


Figure 36.2.2–6 Poorly oriented averaged master surface normals for an irregularly meshed circular surface.

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 master 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" in "Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4). Once the contact direction has been defined, the orientation of the

local tangent plane with respect to the master 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 master surface facet. The position of the anchor point relative to the surrounding nodes on the master surface facet does not change as the master surface deforms.

Load transfer

In a small-sliding analysis the slave node will transfer load to the nodes of the master 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 36.2.2–3 node 103 transmits load to both nodes 2 and 3 on the master 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 slave node slides along its local tangent plane, Abaqus/Explicit does not update the distribution of load transferred by a given slave node to its associated master 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 master surface nodes as sliding occurs, so that no net moment is associated with the contact forces acting on slave and master 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 36.2.2–7 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 slave node 101 in Figure 36.2.2–1 and its master surface **BSURF**. Using the unit normal vectors $\mathbf{N}_{201}$ and $\mathbf{N}_{202}$, the anchor point $\mathbf{X}_0$ was found for slave 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 36.2.2–7, it will continue to transmit load equally to nodes 201 and 202 when, in fact, it really slid off the mesh forming the master 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:

- Slave nodes should slide less than an element length from their corresponding anchor point and still be contacting their local tangent plane. If the master surface is highly curved, the slave 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" in "Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4) to improve the tangent plane orientation.

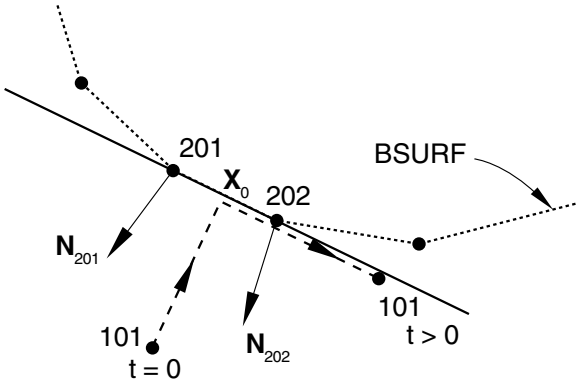


Figure 36.2.2–7 Excessive sliding in a small-sliding contact analysis.

- The rotation and deformation of the master surface should not cause the local tangent planes to become a poor representation of the master surface during the course of the analysis.

Master surface refinement in small-sliding problems

The basic guidelines for pure master-slave 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 master surface.

The smoothly varying master 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 master surface can reorient the local tangent planes such that they become a poor representation of the master surface. Figure 36.2.2–8 shows an example where distortion of the master surface results in such a situation. This problem can be minimized to some extent by using a more refined mesh on the master 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 master surface node remain constant during an infinitesimal-sliding simulation.

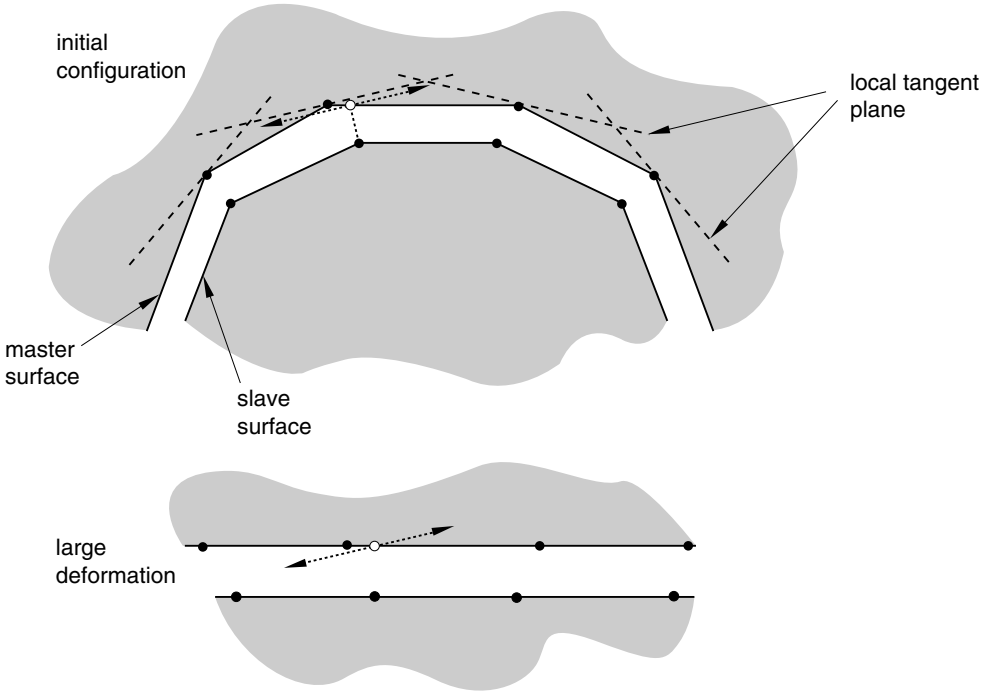


Figure 36.2.2–8 Master surface deformation in a small-sliding contact analysis can cause problems with the local tangent planes.

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: **NIgeom: Off**

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 master 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 master-slave contact pair for discussion purposes. For a balanced master-slave contact pair, the contact search computations are performed twice for each contact pair.

Global contact searches

A global search determines the globally nearest master surface facet for each slave 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 36.2.2–9.

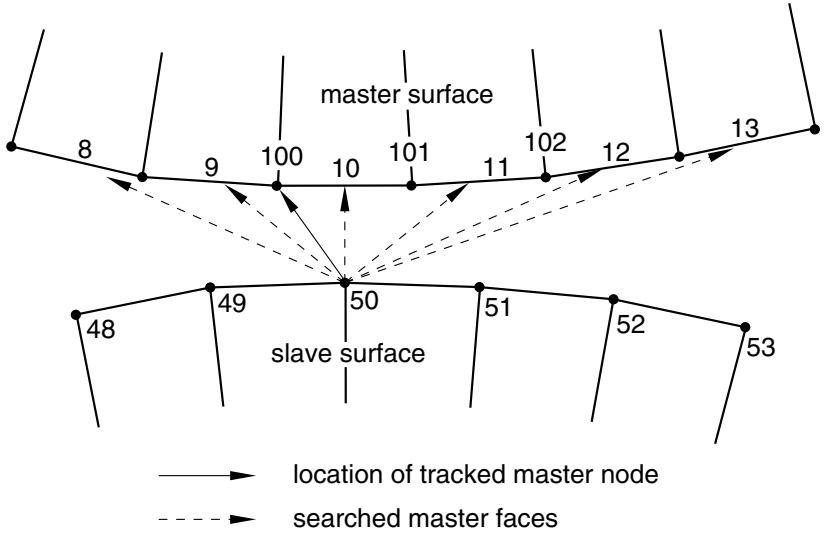


Figure 36.2.2–9 Global search in two dimensions.

The global search computes the distance from node 50 to all of the master surface facets in the same bucket as node 50. It determines that the nearest facet on the master 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 master surface node. This search is conducted for each slave node, comparing each node against all of the facets on the master 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," Section 34.5.5. 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 slave node searches only the facets that are attached to the previously tracked master surface node. Abaqus/Explicit determines which adjacent facet is the nearest to the slave node. It then determines which node on that facet is the closest master surface node to the slave node and updates the tracked master surface node. If the closest master surface node is not the same as the previously tracked master surface node, Abaqus/Explicit performs another iteration of the local search.

In the example shown in Figure 36.2.2–10, node 50 moves as shown during an increment. In the first iteration of the search Abaqus/Explicit finds that the master surface facet on element 10 is still the closest facet of those attached to node 100 but that node 101 is now the tracked master 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 master surface node is 102. Another iteration is performed because the identity of the tracked master 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 master surface node for slave 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," Section 34.5.5.

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," Section 34.5.5). If several facets that are unconnected to each other are found to be near a slave 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.

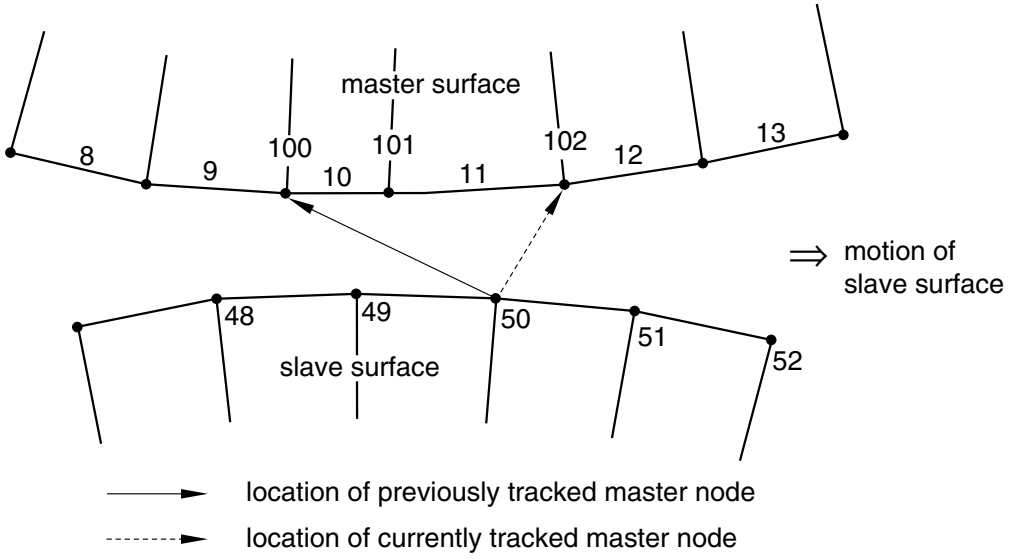


Figure 36.2.2–10 Local search in two dimensions.

Small-sliding (or infinitesimal-sliding) tracking approach

When the small-sliding or infinitesimal-sliding contact approach is invoked (see "Sliding formulation" in "Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2), Abaqus/Explicit performs a single global search at the beginning of the first step to determine the globally nearest master surface facet for each slave 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 slave nodes that do not project onto any master 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.

36.2.3 CONTACT CONSTRAINT ENFORCEMENT METHODS IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Defining general contact interactions in Abaqus/Explicit," Section 34.4.1
- "Defining contact pairs in Abaqus/Explicit," Section 34.5.1
- *CONTACT
- *CONTACT PAIR
- "Specifying master-slave assignments for general contact," Section 15.13.5 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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 master-slave contact pair.

Kinematic enforcement of contact conditions in a pure master-slave 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 slave nodes in the predicted configuration penetrate the master surfaces. The depth of each slave 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 slave node to exactly contact the master surface. The next step depends on the type of master surface used.

- When the master surface is formed by element faces, the resisting forces of all the slave nodes are distributed to the nodes on the master surface. The mass of each contacting slave node is also distributed to the master 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 master surface nodes. Acceleration corrections for the slave nodes are then determined using the predicted penetration for each node, the time increment, and the acceleration corrections for the master 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 master surface, the resisting forces of all slave nodes are applied as generalized forces on the associated rigid body. The mass of each contacting slave 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 master surface. Acceleration corrections for the slave nodes are then determined by the corrected motion of the master surface.

When using hard kinematic contact, it is still possible with the pure master-slave algorithm for the master surface to penetrate the slave surface in the corrected configuration (see Figure 36.2.3–1).

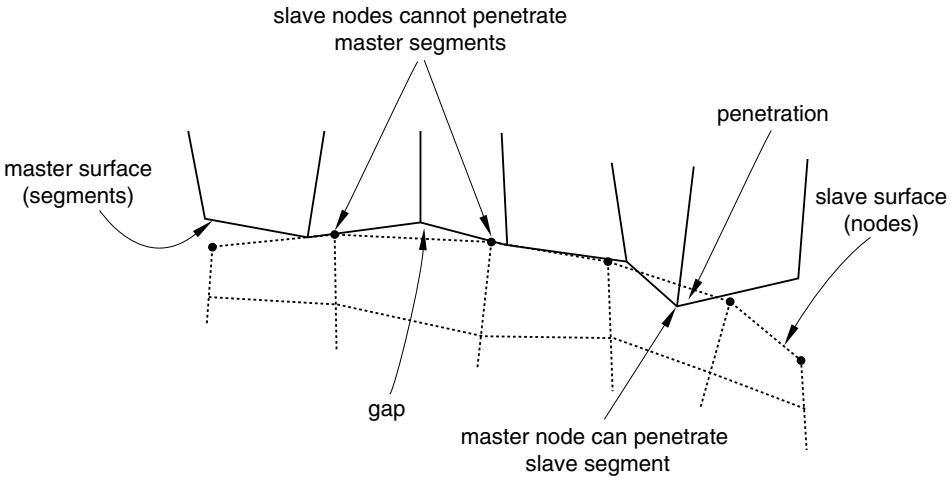


Figure 36.2.3–1 Master surface penetrations into the slave surface of a pure master-slave contact pair due to coarse discretization.

Using a sufficiently refined mesh on the slave surface will minimize such penetrations. Softened kinematic contact will allow penetrations since corrections are made to satisfy the pressure-overclosure relationship at the slave-nodes, not the condition of zero penetration.

Kinematic enforcement of contact conditions in a balanced master-slave contact pair

The kinematic contact algorithm for a balanced master-slave contact pair applies acceleration corrections that are linear combinations of pure master-slave corrections calculated in exactly the same manner as outlined above. One set of corrections is calculated considering one surface as the master surface, and the other corrections are calculated considering that same surface as the slave 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" in "Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2). The default for balanced master-slave 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 master-slave contact pair that uses hard kinematic contact. Both master-slave 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 36.2.3–2 to Figure 36.2.3–5.

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 master-slave kinematic formulation to avoid these inaccuracies.

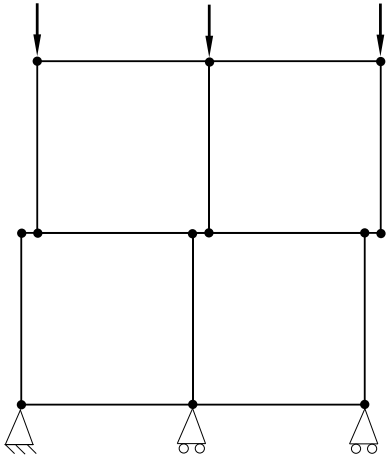


Figure 36.2.3–2 Effect of second contact corrections; initial configuration.

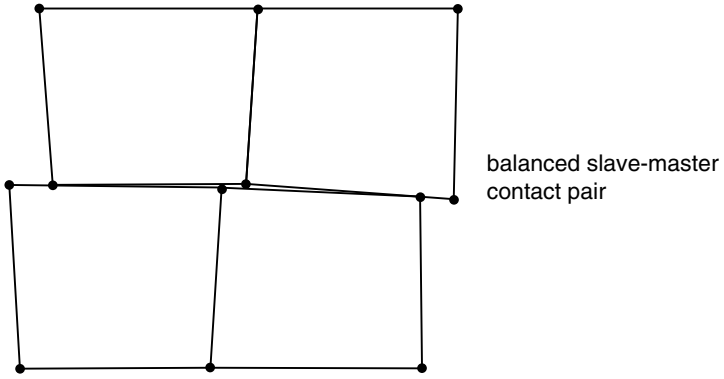


Figure 36.2.3–3 Final configuration when the second contact correction is used.

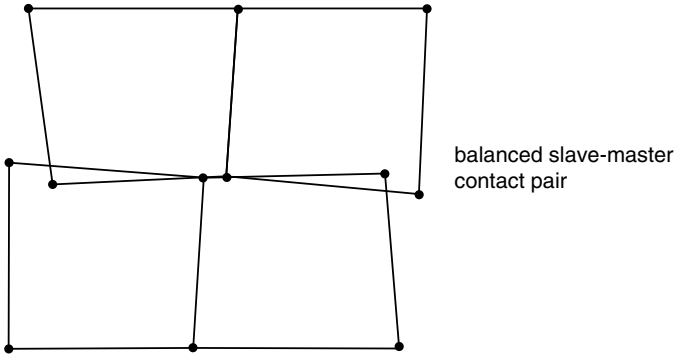


Figure 36.2.3–4 Final configuration if the second contact correction were to be omitted.

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 36.2.3–6. 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.

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

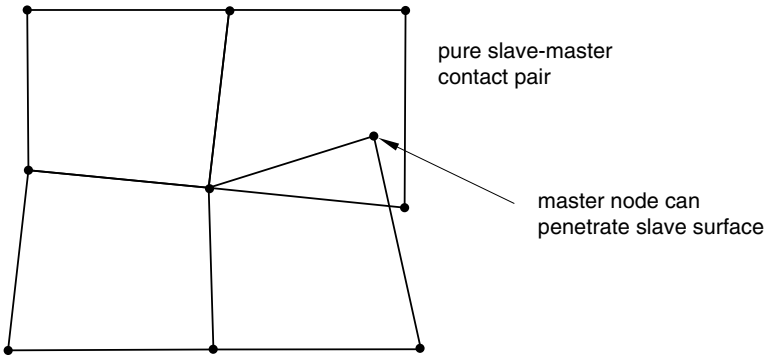


Figure 36.2.3–5 Final configuration when a pure master-slave contact pair is used. The master surface is defined on the bottom elements.

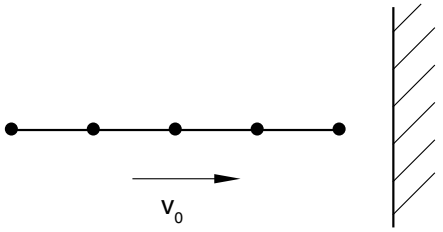


Figure 36.2.3–6 Beam impacting a fixed rigid wall.

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," Section 35.1.5).

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 master-slave surface weighting

The penalty contact algorithm searches for slave 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 slave nodes to oppose the penetration, while equal and opposite forces act on the master surface at the penetration point. The master surface contact forces are distributed to the nodes of the master faces being penetrated. For node-to-analytical rigid surface contact, forces that are a function of the penetration distance are applied to the slave 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 master-slave kinematic contact algorithm, there is no resistance to master surface nodes penetrating slave surface faces with the pure master-slave penalty contact algorithm. Using a sufficiently refined mesh on the slave surface will help correct this problem.

Penalty enforcement of contact conditions for balanced master-slave surface weighting

The penalty contact algorithm for balanced master-slave contact surfaces computes contact forces that are linear combinations of pure master-slave forces calculated in the manner outlined above. One set of forces is calculated considering one surface as the master surface, and the other forces are calculated considering that same surface as the slave 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," Section 36.2.1, and "Contact formulations for contact pairs in Abaqus/Explicit," Section 36.2.2). The default for balanced master-slave contact pairs and general contact is to weight each of the two sets of forces equally.

Scaling the penalty stiffness

The "spring" stiffness that relates the contact force to the penetration distance 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," Section 1.1.11 of the Abaqus Benchmarks Manual.

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 penalty stiffnesses, as described in "Contact controls for general contact in Abaqus/Explicit," Section 34.4.5, and "Contact controls for contact pairs in Abaqus/Explicit," Section 34.5.5. 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 master 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" in "Common difficulties associated with contact modeling using contact pairs in Abaqus/Explicit," Section 37.2.2). 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," Section 35.1.2), 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 36.2.3–1). 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 36.2.3–1 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

37. Contact Difficulties and Diagnostics

Resolving contact difficulties in Abaqus/Standard 37.1
Resolving contact difficulties in Abaqus/Explicit 37.2

37.1 Resolving contact difficulties in Abaqus/Standard

- "Contact diagnostics in an Abaqus/Standard analysis," Section 37.1.1
- "Common difficulties associated with contact modeling in Abaqus/Standard," Section 37.1.2

37.1.1 CONTACT DIAGNOSTICS IN AN Abaqus/Standard ANALYSIS

Products: Abaqus/Standard Abaqus/CAE

References

- "Output to the data and results files," Section 4.1.2
- "Defining general contact interactions in Abaqus/Standard," Section 34.2.1
- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- "Contact formulations in Abaqus/Standard," Section 36.1.1
- *CONTACT PRINT
- *PREPRINT
- *PRINT
- Chapter 41, "Viewing diagnostic output," of the Abaqus/CAE User's Manual

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 four 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 "Controlling initial contact status in Abaqus/Standard," Section 34.2.4, and "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5) or to remove overclosures or gaps between surfaces of surface-based tie constraints (see "Mesh tie constraints," Section 33.3.1). 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," Section 4.2.1) 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, Abaqus/Explicit, and Abaqus/CFD execution," Section 3.2.2). The data check creates an output database and calculates the variable COPEN (contact opening) on each slave 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 slave surface, the internally generated contact element number associated with each slave 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:

- slave and master surface names;
- interaction property;

- value of h_{crit} (see "Controlling the increment size based on penetration distance in unconverged iterations" in "Common difficulties associated with contact modeling in Abaqus/Standard," Section 37.1.2);
- degree of smoothing on the master surface (see "Smoothing master surfaces for the finite-sliding, node-to-surface formulation" in "Contact formulations in Abaqus/Standard," Section 36.1.1);
- characteristic length used in penetration tolerance calculations (see "Augmented Lagrange method" in "Contact constraint enforcement methods in Abaqus/Standard," Section 36.1.2);
- extension ratio applied to master surface edges (see "Extending master surfaces and slide lines," Section 34.3.8); 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 master surface nodes associated with slave 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 master nodes associated with each slave node. Each row of the table lists a slave node and the master nodes to which the slave node transfers load when in contact with the master surface. The number of nodes in the table indicates whether or not the anchor point for a slave 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 "Contact formulations in Abaqus/Standard," Section 36.1.1.

In the output shown below for a two-dimensional model, slave node 2 has an anchor point at master surface node 101 because it interacts with three master surface nodes. Slave node 1 has an anchor point between nodes 100 and 101. This table also provides a list of slave nodes that did not find an intersection with the master surface. This is important because these nodes have no local tangent plane and, hence, can penetrate the master surface.

```
SMALL SLIDING  NON-RIGID  AX ELEMENT(S)
        INTERNALLY GENERATED FOR SLAVE BLANK AND MASTER SPHERE
         WITH SURFACE INTERACTION INF1

    ELEMENT  SLAVE   MASTER
    NUMBER   NODE(S) NODE(S)

       46       1     101     100
```

```
        47        2      102      101      100
        50        9            NO INTERSECTION
 ***WARNING: 1 SLAVE NODES FOUND NO INTERSECTION WITH A MASTER
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 master surface ("slipping") and when a constraint point in motion stops on the master 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" in "Contact constraint enforcement methods in Abaqus/Standard," Section 36.1.2).

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" in "Procedures: overview," Section 6.1.1). 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" in "Frictional behavior," Section 35.1.5).

For detailed instructions on using the diagnostics tool, see Chapter 41, "Viewing diagnostic output," of the Abaqus/CAE User's Manual. 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" in "Output," Section 4.1.1.

Contact output in the data file

When you request contact output to the data file (see "Surface output from Abaqus/Standard" in "Output to the data and results files," Section 4.1.2), 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 37.1.1–1.

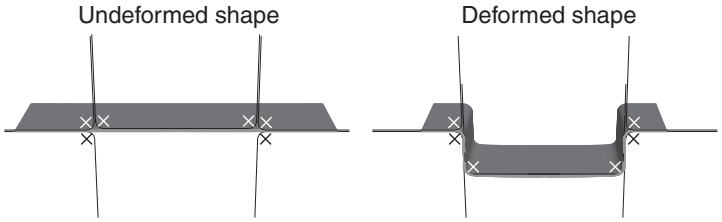


Figure 37.1.1–1 Model for channel-forming example. (The blank has been extruded for visualization purposes.)

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 37.1.1–2).

You can also use the contact conditions to review changes in contact status throughout the forming process. Figure 37.1.1–3 depicts the onset of slipping for two nodes on the blank. 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.

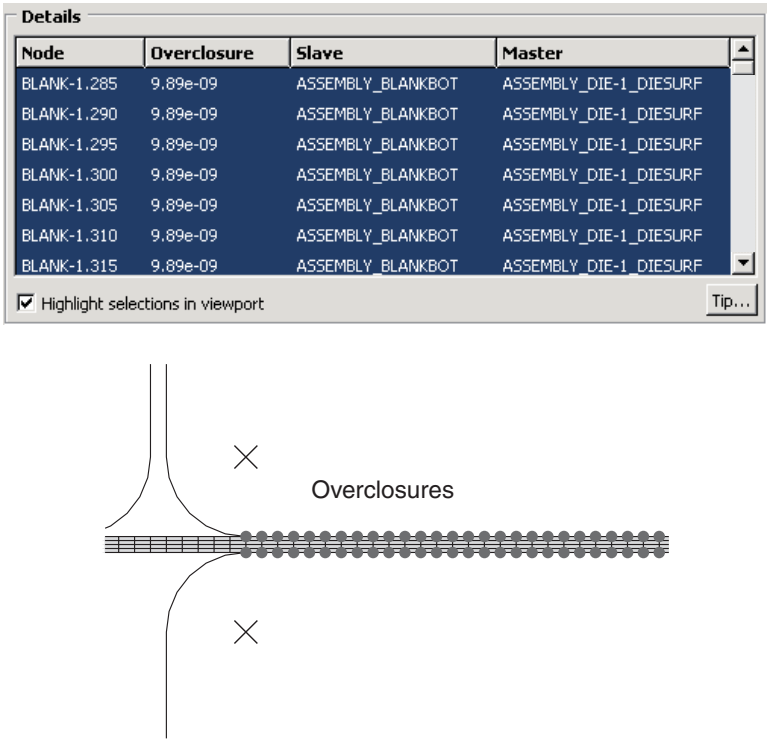


Figure 37.1.1–2 Diagnostics confirming contact conditions between the blank, die, and holder.

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 37.1.1–3:

```
     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 master surface. The SL footnote for node 295 indicates that it is in contact with the master surface (the die) and is

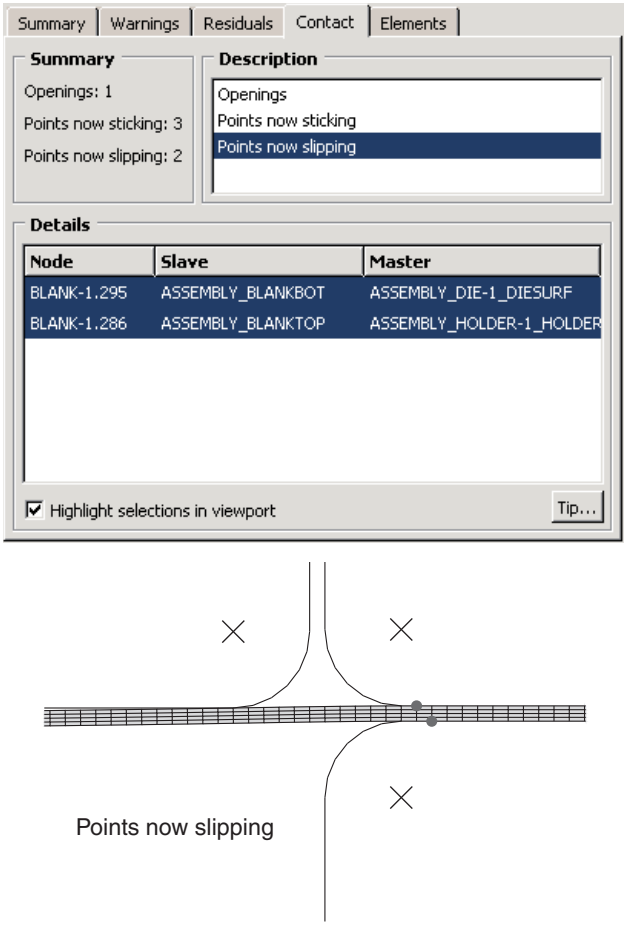


Figure 37.1.1–3 Diagnostics for the onset of slipping.

"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 slave node. Accumulated slip and slip directions are discussed in more detail in "Output of tangential results" in "Defining contact pairs in Abaqus/Standard," Section 34.3.1.

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," Section 37.1.2.

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" in "Procedures: overview," Section 6.1.1), 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 37.1.1–4. 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 37.1.1–4, for example). Techniques for resolving contact chattering problems are discussed in "Excessive iterations in contact simulations" in "Common difficulties associated with contact modeling in Abaqus/Standard," Section 37.1.2.

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 37.1.1–5. 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," Section 7.1.1; "Dashpots," Section 31.2.1; and "Automatic stabilization of rigid body motions in contact problems" in "Adjusting contact controls in

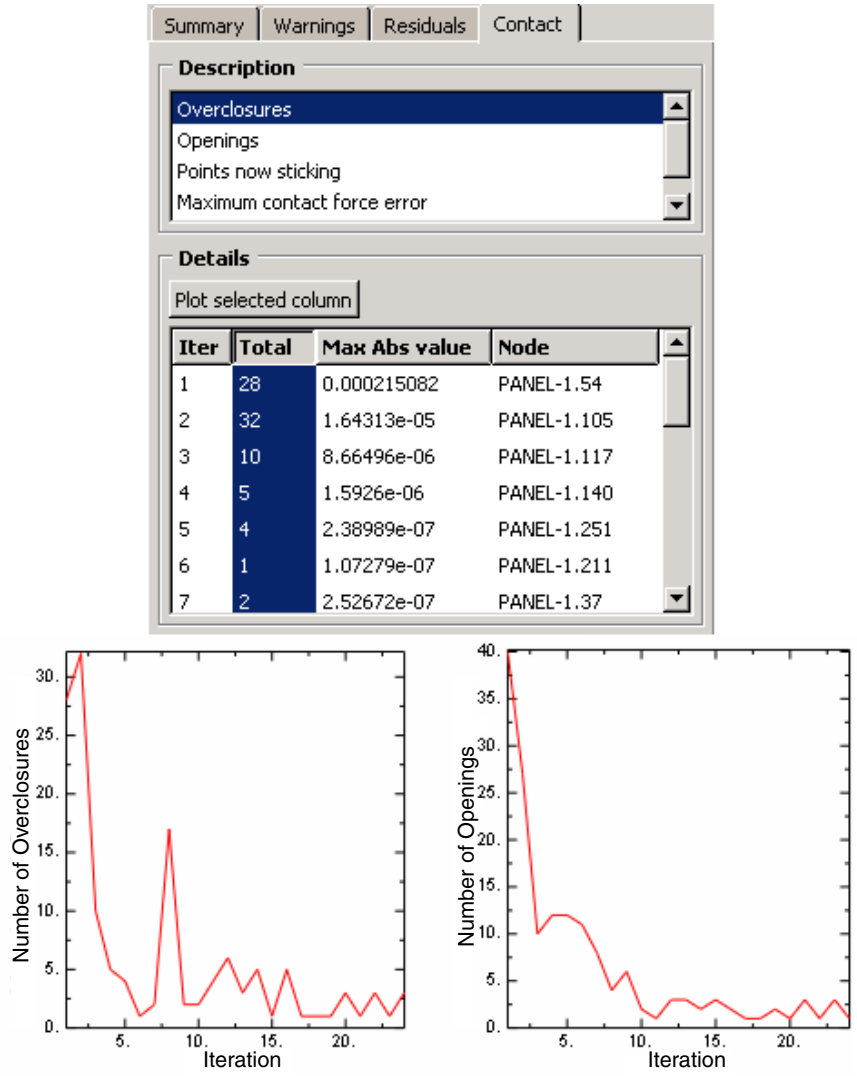


Figure 37.1.1–4 Changes in contact status during an attempt.

Abaqus/Standard," Section 34.3.6). 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 37.1.1–6). If the same node appears

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 37.1.1–5 The overclosure at one constraint point is significantly higher than the overclosures at other constraint points.

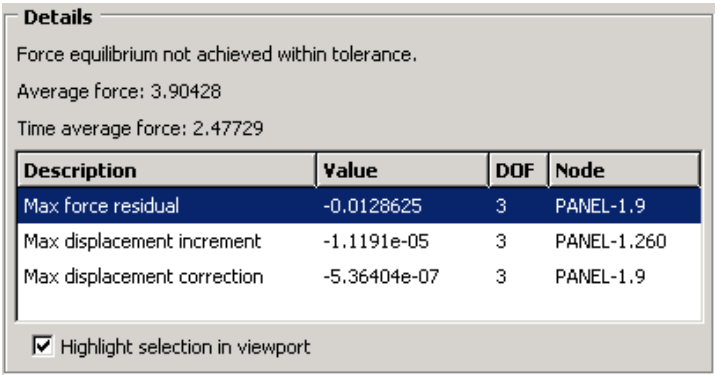


Figure 37.1.1–6 The diagnostics tool reports equilibrium difficulties.

repeatedly as the location of maximum residuals and corrections, investigate the contact conditions around that node. Consider the example in Figure 37.1.1–7. The diagnostics highlight the "problem node" on the perimeter of the slave surface. A closer look in the vicinity of this node reveals that the slave surface mesh is too coarse. Slave nodes along the perimeter of the surface are touching the master surface, but the next row of nodes is "hanging over" the rim of the master surface. If this contact pair uses node-to-surface contact discretization, the master surface can penetrate the slave surface with little resistance between the nodes. Such penetrations can cause the nonconverging force equations seen in the diagnostics.

Any situation in which the master surface is free to penetrate the slave surface can prevent an analysis from converging. Potential solutions include:

- switching the master and slave assignments;
- using surface-to-surface discretization (however, using surface-to-surface discretization without refining a coarse slave mesh may lead to inaccurate stress results, even if the analysis does converge); or
- refining the mesh on the slave surface.

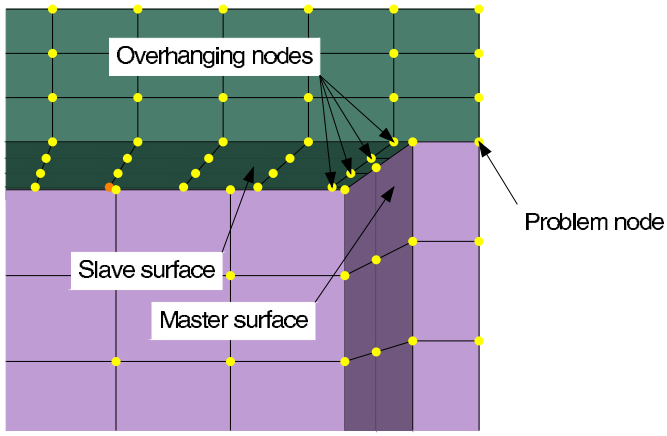


Figure 37.1.1–7 Two surfaces in a region of nonconverging force equations.

37.1.2 COMMON DIFFICULTIES ASSOCIATED WITH CONTACT MODELING IN Abaqus/Standard

Products: Abaqus/Standard Abaqus/CAE

References

- "Defining general contact interactions in Abaqus/Standard," Section 34.2.1
- "Defining contact pairs in Abaqus/Standard," Section 34.3.1
- *CONTACT
- *CONTACT PAIR
- *CONTACT INITIALIZATION DATA
- "Defining general contact," Section 15.13.1 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining surface-to-surface contact," Section 15.13.6 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Using contact and constraint detection," Section 15.16 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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" in "Output," Section 4.1.1). 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 slave surface to ensure that all slave nodes that should initially be in contact with the master surface start out in contact without any penetration (see "Controlling initial contact status in Abaqus/Standard," Section 34.2.4, and "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5). 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 "Defining a precise initial clearance or overclosure for small-sliding contact" in "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5).

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 slave node interacts with a contact plane defined from the finite element approximation of the master surface, as discussed in "Contact formulations in Abaqus/Standard," Section 36.1.1. Abaqus/Standard can define these planes only when each slave node can be projected onto the master surface. Having these slave nodes start the simulation contacting the master surface allows Abaqus/Standard to form the most accurate contact planes for the slave 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" in "Defining contact pairs in Abaqus/Standard," Section 34.3.1. 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," Section 34.3.4), 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," Section 37.1.1, 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, Abaqus/Explicit, and Abaqus/CFD execution," Section 3.2.2) 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 master and slave surfaces of the interaction involved in the initial overclosure is helpful for identifying the cause of an unintended initial overclosure (see "Managing display groups," Section 78.2 of the Abaqus/CAE User's Manual, for a discussion of the display group options). Viewing the surface normals (see "Displaying element and surface normals," Section 55.7 of the Abaqus/CAE User's Manual) may help determine whether unintended overclosures are due to incorrect surface normals.

Overclosures on discontinuous surfaces

Figure 37.1.2–1 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 34.3.1–1 "Orientation considerations for shell-like surfaces" in "Defining contact pairs in Abaqus/Standard," Section 34.3.1, shows which contact formulations allow discontinuous surfaces). The arrows in Figure 37.1.2–1 show the positive normal direction for each surface region. The surface-to-surface contact formulation searches along the slave-surface normal direction (in the positive and negative directions) for potential interaction points on the master 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.

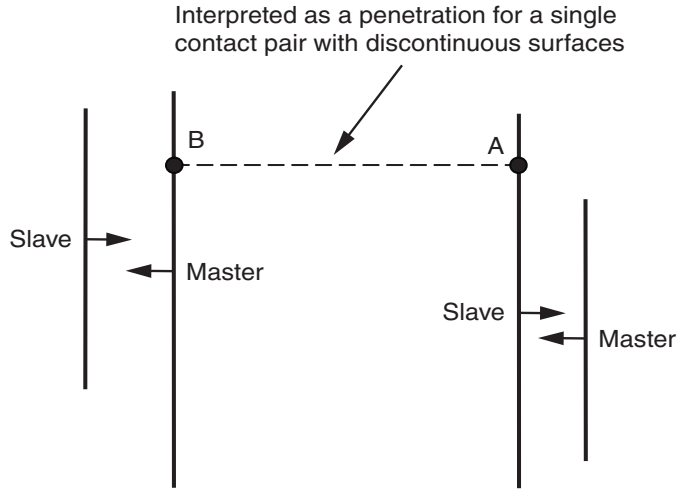


Figure 37.1.2–1 Example of an unintended initial overclosure due to a modeling error involving discontinuous surfaces.

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 master surface should be apparent behind the slave surface (opposite the slave 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 master surface typically corresponds to a local minimum distance between the slave and master 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," Section 34.3.4, 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 "Controlling initial contact status in Abaqus/Standard," Section 34.2.4).

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 "Controlling initial contact status in Abaqus/Standard," Section 34.2.4, and "Adjusting initial surface positions and specifying initial clearances in Abaqus/Standard contact pairs," Section 34.3.5). 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" in "Adjusting contact controls in Abaqus/Standard," Section 34.3.6.

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 master 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 37.1.2–2.

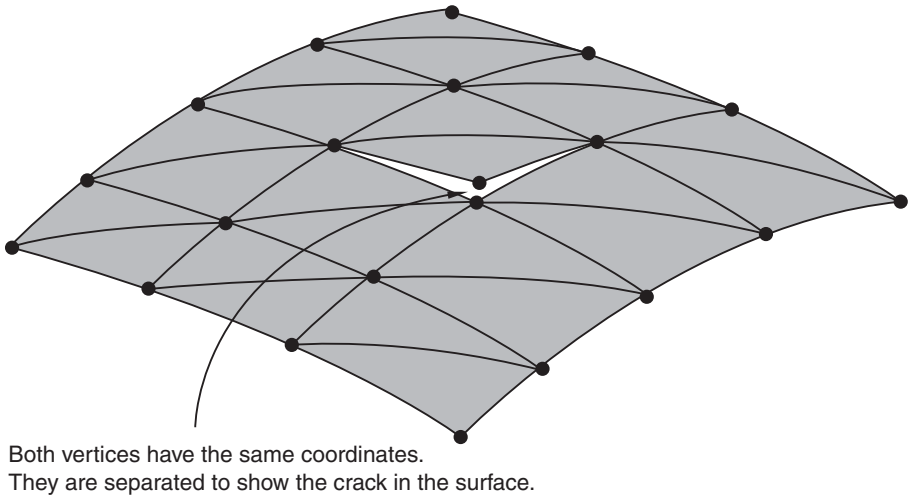


Figure 37.1.2–2 Example of doubly defined surface node.

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 master surface for finite-sliding, node-to-surface contact, a slave node sliding along the surface may fall through this crack and get "stuck" behind the master 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 master surface definition extends far enough to account for all expected motions of the contacting parts. Contact along the perimeter of master surfaces should be avoided with the node-to-surface contact formulation.. Abaqus/Standard assumes that the mating slave surface nodes can fall off the free edge of the master surface, which can cause problems if a slave node wraps around and approaches its mating master surface from behind. Figure 37.1.2–3 illustrates appropriate and inappropriate master surface definitions.

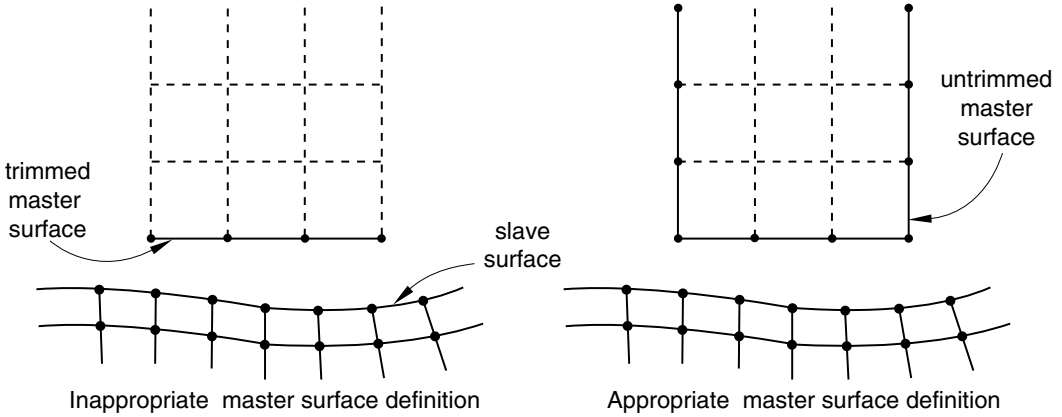


Figure 37.1.2–3 Example of master surface extension.

A slave node that falls off a master 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 slave surface rather than individual slave nodes. Request detailed contact printout to the message (**`.msg`**) file to monitor the history of a slave node that might slide off the master surface (see "The Abaqus/Standard message file" in "Output," Section 4.1.1). The message file output will show the cyclic opening and closing of contact at a slave node, which will indicate where the master surface needs to be modified.

For node-to-surface contact you can extend the master 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 master surfaces and slide lines," Section 34.3.8, for details.

Falling off small-sliding master surfaces

Falling off the edge of a master surface in small-sliding contact problems is not an issue since slave nodes do not slide on the actual surface of the model. Instead, each slave node interacts with a flat, infinite contact plane. This plane is associated with the set of master surface nodes that are closest to

the slave node in the undeformed configuration. For details about small-sliding contact, see "Contact formulations in Abaqus/Standard," Section 36.1.1.

Falling off surfaces modeled with interface elements

Falling off the edge of a surface modeled with interface elements is not an issue since the slave 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 slave surfaces

When a coarsely meshed surface is used as a slave surface for node-to-surface contact, the master surface nodes can grossly penetrate the slave surface without resistance (see Figure 37.1.2–4). This situation is common when nonmatching meshes come into contact. Refining the slave surface tends to alleviate this problem.

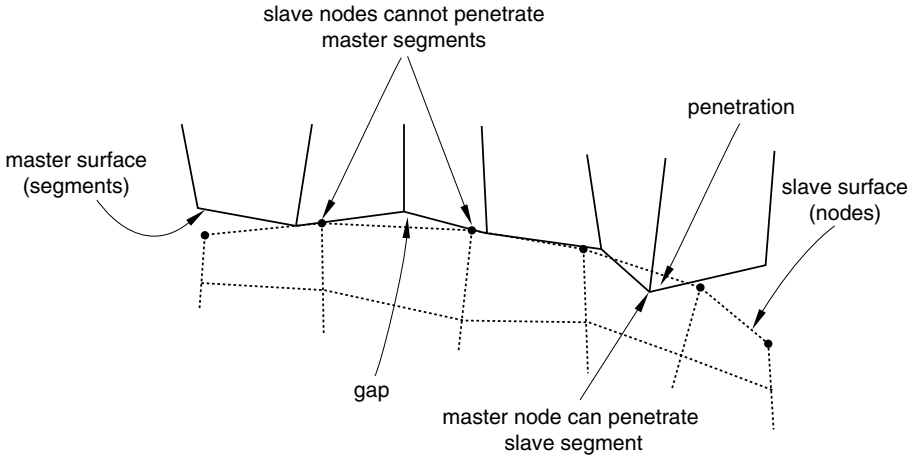


Figure 37.1.2–4 Master surface penetrations into the slave surface due to a coarse mesh of the slave surface for node-to-surface contact.

Surface-to-surface contact will generally resist penetrations of master nodes into a coarse slave surface; however, this formulation can add significant computational expense if the slave mesh is significantly coarser than the master mesh (see "Contact formulations in Abaqus/Standard," Section 36.1.1, 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 37.1.2–5.

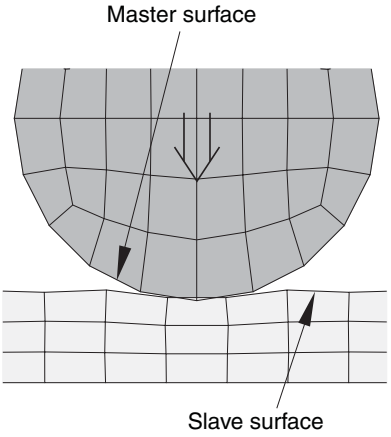


Figure 37.1.2–5 The master surface contacts the slave surface at a single element face.

The results from such an interaction are unreliable and generally unrealistic. If the model in Figure 37.1.2–5 uses node-to-surface contact, the master surface penetrates the slave surface without resistance until it encounters a slave node, as discussed above. If the master and slave designations are reversed, the contact constraint is applied at a single slave 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 master surfaces and small-sliding contact

Coarsely meshed, curved master surfaces in small-sliding simulations can lead to unacceptable solution accuracy due to the approximate nature of the “master planes.” Using a more refined mesh to define the master 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," Section 36.1.1, for a discussion of these contact formulations).

Some second-order element types are not well-suited for underlying the slave 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 37.1.2–6, 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.

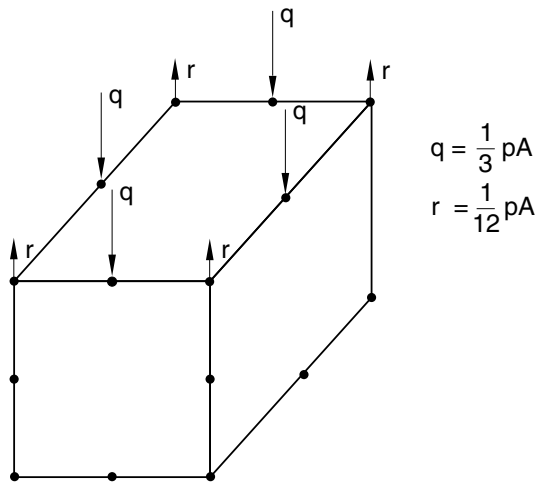


Figure 37.1.2–6 Equivalent nodal loads produced by a constant pressure on the second-order element face in "hard" contact simulations.

Abaqus/Standard bases important decisions for the node-to-surface contact formulation on contact forces acting on individual slave 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 (serendipity

elements) that form a slave 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," Section 36.1.2). 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.

Second-order tetrahedral elements (C3D10 and C3D10I) have zero contact force at their corner nodes. This combination of second-order triangular slave 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" in "Procedures: overview," Section 6.1.1. 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 slave 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 master surface is highly curved. The default value of h_{crit} may sometimes lead to situations as shown in Figure 37.1.2–7. During the iterative solution process a slave node initially at point *a* may move to point *b*, penetrating the master surface with overclosure *h* less than h_{crit}. Abaqus/Standard may attempt to move the slave node to point *c* on the master 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.

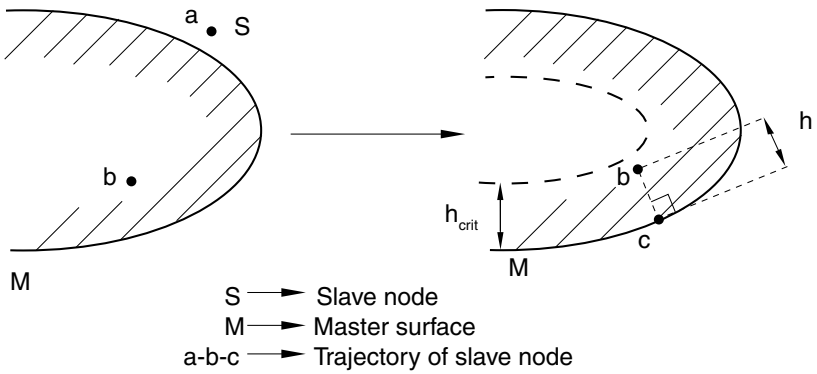


Figure 37.1.2–7 Effect of the critical penetration distance on a highly curved master surface.

- 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 slave 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 slave surface vary greatly.
- Models in which the slave 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," Section 35.1.2).

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," Section 36.1.2).

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 master and parts as slave. The same unpredictability makes it impossible to determine a priori which side will be the master and which side the slave. Therefore, Abaqus/Standard uses a symmetric contact model: every single node of the surface can be a slave node and can simultaneously belong to master segments with respect to all other nodes.

Because each surface is acting as both a slave and a master, the results of symmetric contact analyses can be confusing and inconsistent. These difficulties are discussed more fully in "Using symmetric master-slave contact pairs to improve contact modeling" in "Defining contact pairs in Abaqus/Standard," Section 34.3.1.

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," Section 33.6.1, 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," Section 36.1.2.

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 Chapter 41, "Viewing diagnostic output," of the Abaqus/CAE User's Manual). 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 master surface and share some slave nodes (perhaps along a common edge of two slave surfaces). Overconstraints will occur at the common slave 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," Section 34.3.3, 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," Section 36.1.1) 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 slave nodes, while surface-to-surface discretization computes penetrations in an average sense over a finite region. For example, when a slave surface slides across a convex portion of a master surface, the slave surface will tend to ride a bit higher with surface-to-surface discretization than with node-to-surface discretization, as shown in Figure 37.1.2–8 (the opposite is true at a concave portion of a master surface). Figure 37.1.2–9 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.

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.

Contact at a single point

Figure 37.1.2–10 shows an example in which a circular rigid body is pushed into a deformable body. In the initial configuration shown, the two bodies touch at a single point, which corresponds to a slave node location. The following scenarios are likely for respective analyses of this model with node-to-surface and surface-to-surface discretization:

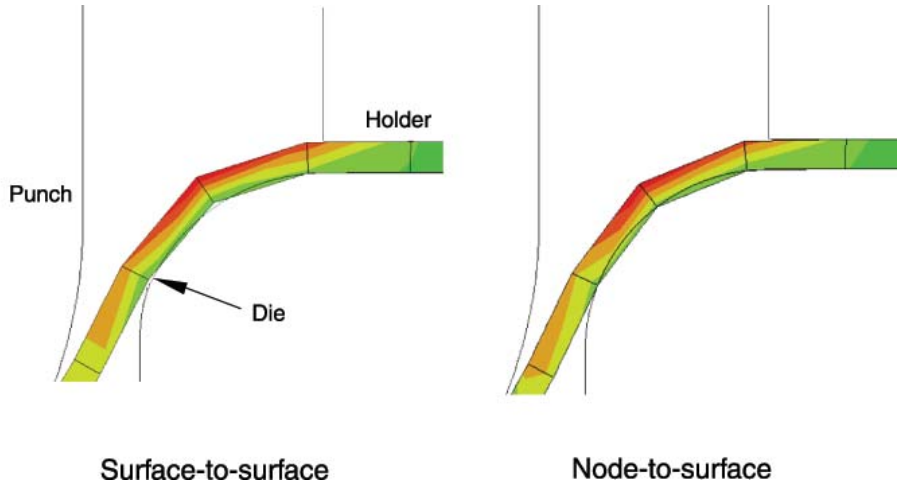


Figure 37.1.2–8 Comparison of contact discretizations in an example with convex curvature in the master surface (forming application).

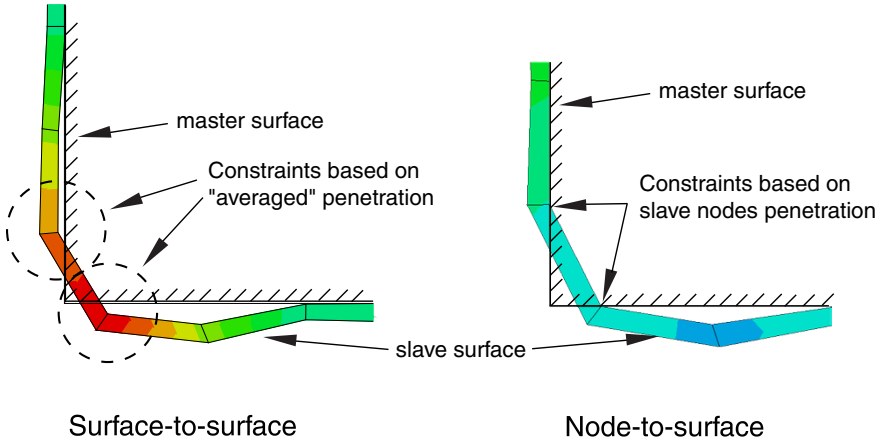


Figure 37.1.2–9 Comparison of contact discretizations in an example with a relatively flexible slave surface wrapping around a corner of a master surface.

- 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.

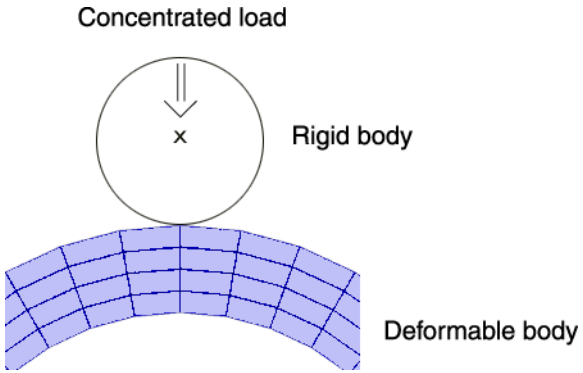


Figure 37.1.2–10 Example with two bodies initially touching at a single point.

- 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 slave 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 master surface normal, whereas surface-to-surface discretization uses a contact normal direction based on the slave surface normal (averaged over a region nearby the slave node). For most active contact definitions the slave and master surfaces are nearly parallel, so the master and slave 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 slave surface as the overclosures are resolved. This tangential motion may have undesirable effects on an analysis. See "Controlling initial contact status in Abaqus/Standard," Section 34.2.4, and "Modeling contact interference fits in Abaqus/Standard," Section 34.3.4, 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 slave and master 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 master surface; for surface-to-surface discretization, the opening distance that is reported corresponds to the distance from the slave surface to the master surface along the slave normal direction. The opening distance for surface-to-surface discretization is undefined if a line emanating from the slave surface in the slave normal direction does not intersect the master surface (as discussed in "Using the small-sliding tracking approach" in "Contact formulations in Abaqus/Standard," Section 36.1.1, 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 37.1.2–11, the slave surface is on the "outer" body (i.e., the body with a reentrant corner). With node-to-surface discretization a single constraint acts at the corner slave node in the "average" normal direction of the master 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 37.1.2–11, resulting in more stable contact behavior.

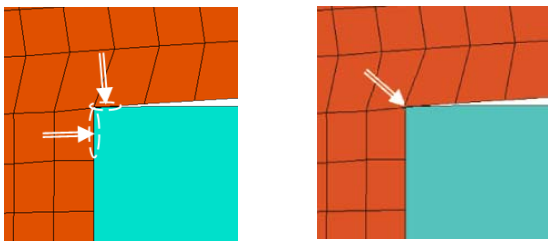


Figure 37.1.2–11 Comparison of contact formulations in an example with abutting surfaces having respective interior and exterior corners.

37.2 Resolving contact difficulties in Abaqus/Explicit

- "Contact diagnositcs in an Abaqus/Explicit analysis," Section 37.2.1
- "Common difficulties associated with contact modeling using contact pairs in Abaqus/Explicit," Section 37.2.2

37.2.1 CONTACT DIAGNOSITCS IN AN Abaqus/Explicit ANALYSIS

Products: Abaqus/Explicit Abaqus/CAE

References

- "Output to the data and results files," Section 4.1.2
- "Contact interaction analysis: overview," Section 34.1.1
- *DIAGNOSTICS
- Chapter 41, "Viewing diagnostic output," of the Abaqus/CAE User's Manual

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 "Controlling initial contact status for general contact in Abaqus/Explicit," Section 34.4.4, and "Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4). 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. 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, Abaqus/Explicit, and Abaqus/CFD execution," Section 3.2.2).

Visualizing the precise initial clearances for small-sliding contact pairs

Abaqus/Explicit does not adjust the coordinates of the slave surface when precise initial clearances are specified for small-sliding contact pairs (see "Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4). 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 slave surface initially penetrates a double-sided master surface by a distance greater than the master surface's thickness, the severely overclosed slave nodes will see the back side of the master surface as the appropriate contact force direction. These slave nodes in these crossed surfaces effectively become trapped behind the master surface. This issue is discussed in more detail in "Controlling initial contact status for general contact in Abaqus/Explicit," Section 34.4.4, and "Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4.

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 slave nodes (connected by a facet edge) are detected on opposite sides of a master surface. No such warning is issued for node-based surface nodes on opposite sides of a master surface, because adjacency cannot be determined among the node-based surface nodes. In some cases involving corners of master surfaces this warning message may be issued even though adjacent slave nodes are really on the same side of a master 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," Section 36.2.3, 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," Section 34.5.5). 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.

37.2.2 COMMON DIFFICULTIES ASSOCIATED WITH CONTACT MODELING USING CONTACT PAIRS IN Abaqus/Explicit

Products: Abaqus/Explicit Abaqus/CAE

References

- "Defining contact pairs in Abaqus/Explicit," Section 34.5.1
- *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 "Defining general contact interactions in Abaqus/Explicit," Section 34.4.1, 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 master 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 37.2.2–1.

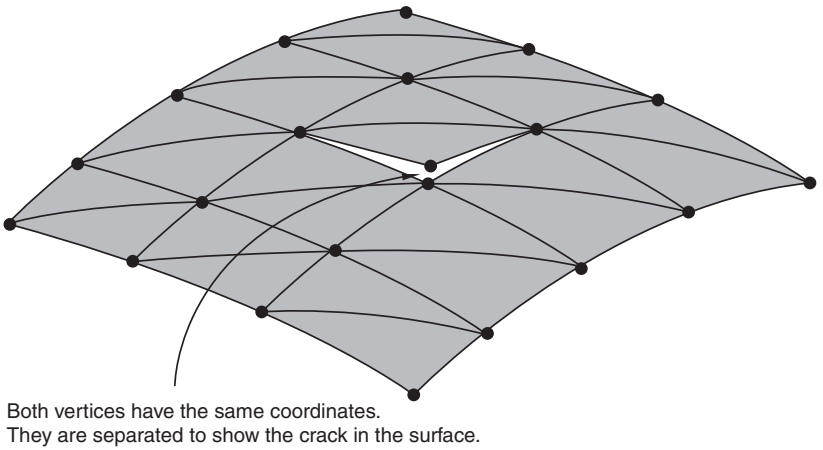


Figure 37.2.2–1 Example of doubly defined surface node.

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 37.2.2–2 shows a two-dimensional model of a simple connection between two parts.

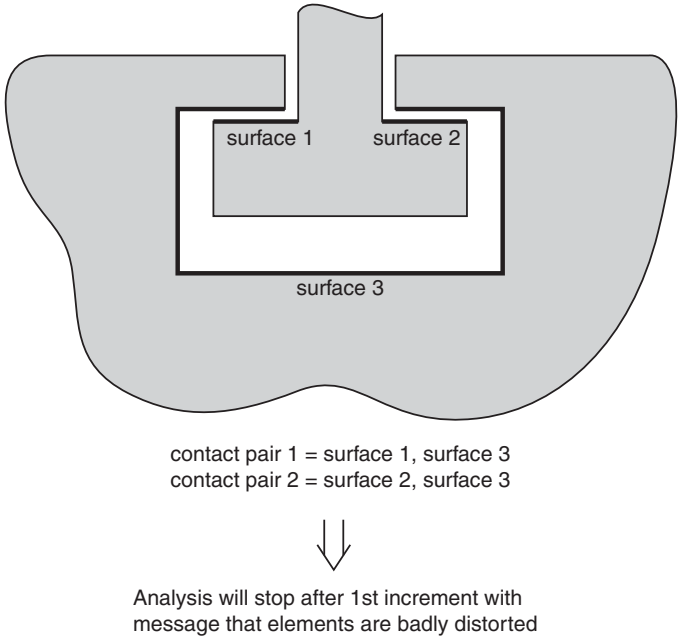


Figure 37.2.2–2 Surface definitions that are inadequate for the desired contact conditions.

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 37.2.2–3.

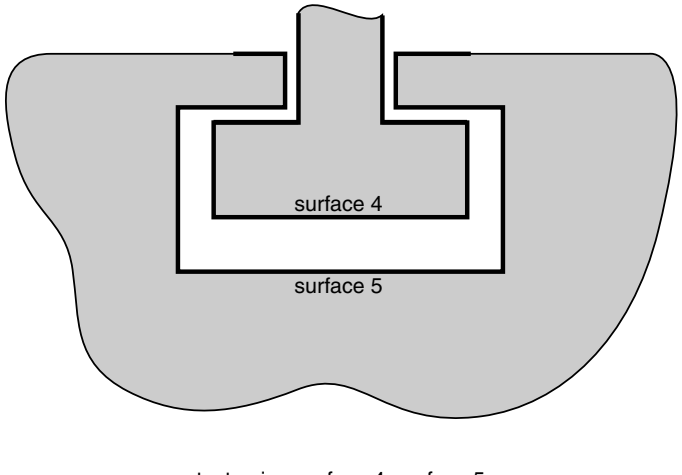


Figure 37.2.2–3 Surface definitions that are adequate for the desired contact conditions.

The surfaces shown in that figure are suitable for the desired contact definition. Other solutions, such as using a pure master-slave 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 slave surface in a pure master-slave contact pair with hard surface behavior, an inaccurate solution may be produced as a result of the gross penetration of the master surface into the slave surface. This situation is shown in Figure 37.2.2–4. This problem can be minimized if the contact pair can be switched to a balanced master-slave contact pair. However, some contact pairs in Abaqus/Explicit must always use a pure master-slave formulation. In these cases the only solution to gross penetration is to refine the slave surface.

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 slave 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

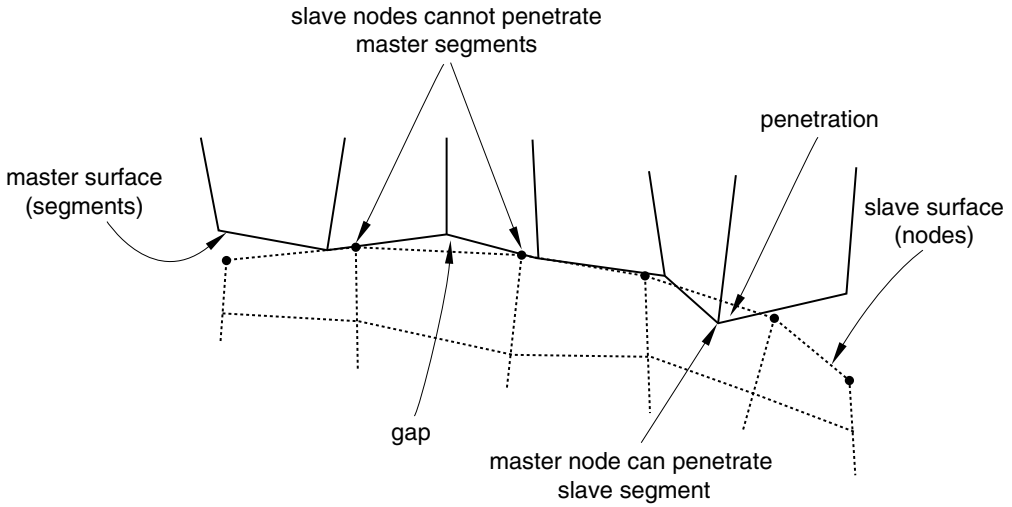


Figure 37.2.2–4 Master surface penetrations into the slave surface due to coarse discretization.

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," Section 2.3.4.

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 master 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 slave 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 master 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 master surfaces (or increasing the mass of the pinched nodes) will also increase the decay rate, although a high ratio of slave mass to master 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 slave node in two or more simultaneous, kinematic contact constraints, the resulting contact corrections may be erroneous, possibly causing the analysis to abort 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 slave in two contact pairs and there is a possibility of simultaneous contacts for individual slave 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 slave surface and master 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 master-slave contact, the slave 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 slave surfaces so that the shared nodes will not act as slave nodes in the contact pair. However, the shared nodes will still be used in the definition of a master 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 37.2.2–5.

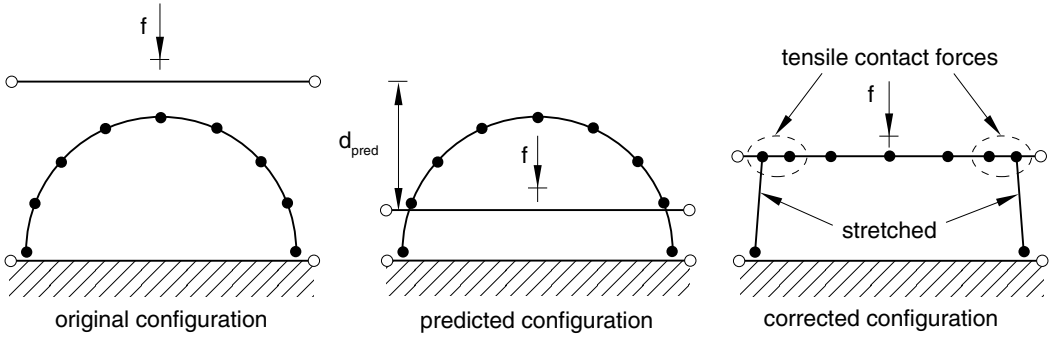


Figure 37.2.2–5 Undesirable numerical behavior of contact algorithm resulting from small rigid body mass.

With hard kinematic contact each slave node that is penetrating its master surface in the predicted configuration will be brought to the position of its tracked point on the master surface in the corrected configuration, which, in this example, generates tensile contact forces at the outer slave 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 master nodes are orders of magnitude less than those of the slave nodes. This problem can often be avoided in such cases by using the pure master-slave algorithm with the master 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 master-slave contact constraint with finite sliding is defined near a symmetry plane in the master surface, the corner slave node (node A in Figure 37.2.2–6) can, under some circumstances, slide freely along the symmetry plane without experiencing contact. If the master surface wraps around the corner (node 1), the slave node A may "track" on the master segment (1–6) on the symmetry plane, rather than on master segment (1–2). The result may be an inaccurate representation of the contact constraint as shown by the shaded area.

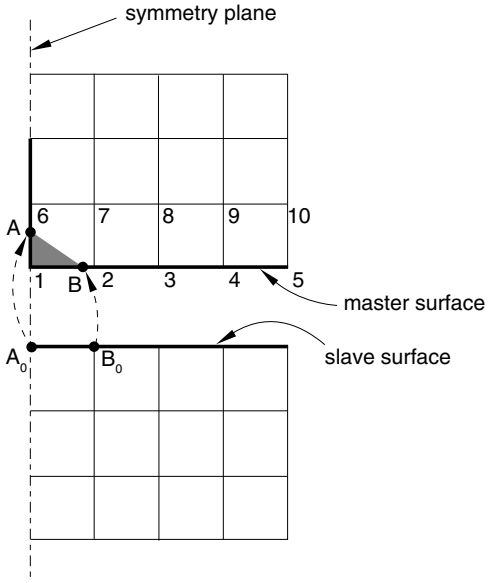


Figure 37.2.2–6 Contact near a symmetry plane. The master surface is wrapped around the corner.

If the master surface does not wrap around the corner (node 1 in Figure 37.2.2–7), the contact logic may give different results depending on how the symmetry boundary conditions have been defined for the master node 1 on the symmetry plane. If the symmetry boundary conditions on the master node are specified using boundary "type" format (i.e., XSYMM, YSYMM, or ZSYMM—see "Boundary conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.3.1), the master surface is effectively extended beyond the symmetry plane (Figure 37.2.2–7); thus, the slave node A will be detected as a "penetrated" node (penetrated by distance a). Therefore, a correcting force would be applied on slave node A to push it below the master surface.

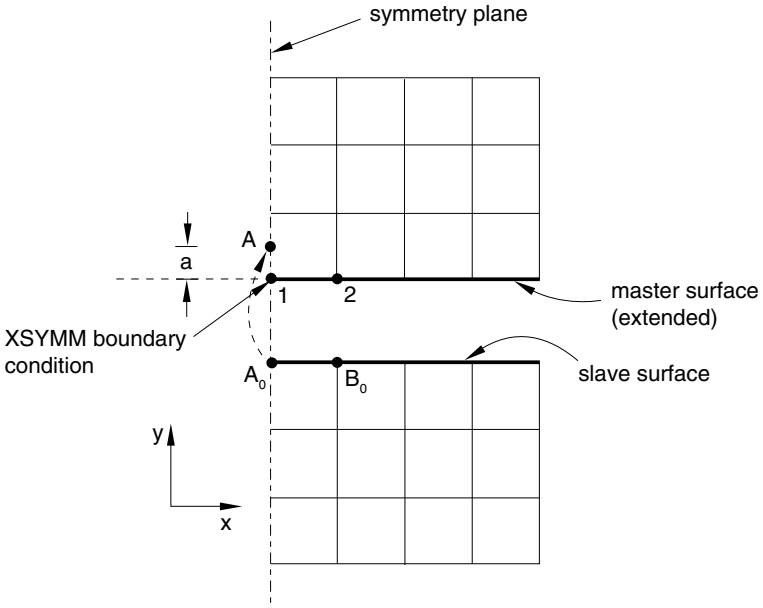


Figure 37.2.2–7 The master surface is extended across the symmetry plane because the symmetry boundary condition at node 1 is specified using boundary type XSYMM.

If the symmetry boundary conditions on the master node 1 are specified using "direct" format (i.e., specifying the components of translations and rotations that are fixed), the master surface is not extended beyond the symmetry plane (Figure 37.2.2–8) and it is possible that contact will not be enforced correctly.

To ensure proper enforcement of finite-sliding contact near symmetry planes, use balanced master-slave contact or use pure master-slave contact without extending the surface onto the symmetry plane and use symmetry "type" boundary conditions on the perimeter of the master 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," Section 36.2.2.

Specifying initial clearance values precisely

You can define initial clearances and contact directions precisely for the nodes on the slave surface (see "Specifying initial clearance values precisely" in "Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4). The initial clearance or overclosure value calculated at every slave node based on the coordinates of the slave node and the master surface is overwritten by the value that you specify; the coordinates of the slave 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

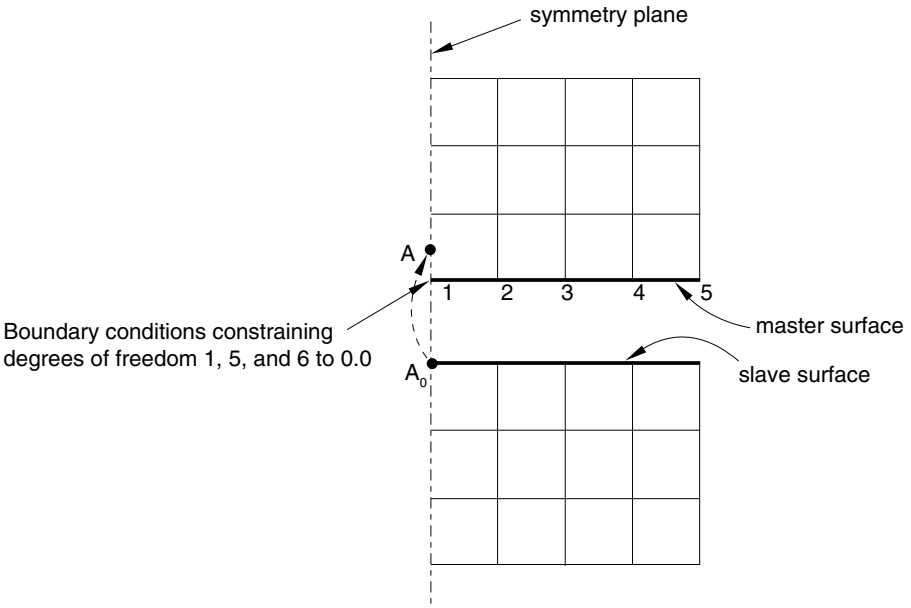


Figure 37.2.2–8 The master surface is not extended across the symmetry plane because the symmetry boundary conditions at node 1 are specified using direct format.

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," Section 36.2.2).

When the balanced-master slave 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 master surfaces will be ignored.

Visualizing the precise initial clearances for small-sliding contact pairs

Abaqus/Explicit does not adjust the coordinates of the slave surface when precise initial clearances are specified for small-sliding contact pairs (see "Adjusting initial surface positions and specifying initial clearances for contact pairs in Abaqus/Explicit," Section 34.5.4). 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.

38. Contact Elements in Abaqus/Standard

Contact modeling with elements 38.1
Gap contact elements 38.2
Tube-to-tube contact elements 38.3
Slide line contact elements 38.4
Rigid surface contact elements 38.5

38.1 Contact modeling with elements

- “Contact modeling with elements,” Section 38.1.1

38.1.1 CONTACT MODELING WITH ELEMENTS

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 (Chapter 34, "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," Section 38.2.1). 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," Section 38.3.1) 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," Section 38.4.1) 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," Section 38.5.1). For example, rigid surface contact elements might be used to model the contact between a rubber seal and a much stiffer structure.

38.2 Gap contact elements

- “Gap contact elements,” Section 38.2.1
- “Gap element library,” Section 38.2.2

38.2.1 GAP CONTACT ELEMENTS

Product: Abaqus/Standard

References

- "Gap element library," Section 38.2.2
- *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 Chapter 34, "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

$$h = d + \mathbf{n} \cdot (\mathbf{u}^2 - \mathbf{u}^1),$$

where $\mathbf{u}^1$ and $\mathbf{u}^2$ are the total displacements at the first and the second node forming the GAPUNI element. Figure 38.2.1–1 shows the configuration of the GAPUNI element. When h becomes negative, the gap contact element is closed and the constraint $h = 0$ is imposed.

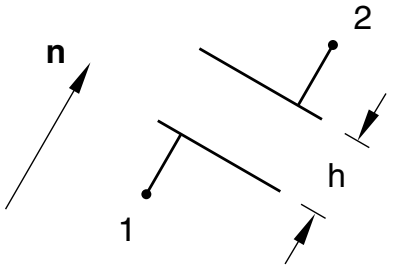


Figure 38.2.1–1 GAPUNI and GAPUNIT contact elements.

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," Section 1.2.2). 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 38.2.1–2.

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 $d = 0$ 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 38.2.1–2). 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.

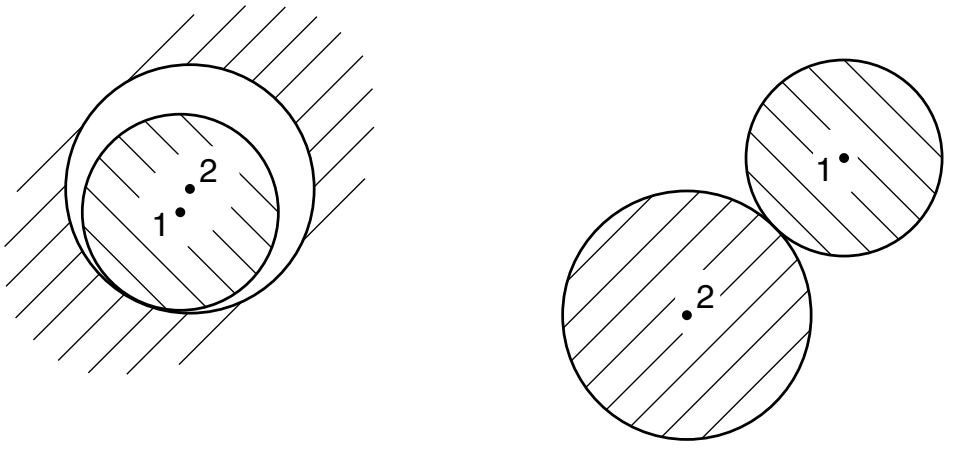


Figure 38.2.1–2 Gap clearance for GAPCYL/GAPSPHER contact elements.

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

$$h = d - |\overline{\mathbf{x}}^2 - \overline{\mathbf{x}}^1| \qquad \text{with} \qquad \overline{\mathbf{x}}^N = \mathbf{x}^N - \mathbf{a}\,(\mathbf{a} \cdot \mathbf{x}^N),$$

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 38.2.1–2). 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

$$h = \left|\overline{\mathbf{x}}^2 - \overline{\mathbf{x}}^1\right| - |d| \qquad \text{with} \qquad \overline{\mathbf{x}}^N = \mathbf{x}^N - \mathbf{a}\,(\mathbf{a}\cdot\mathbf{x}^N).$$

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," Section 1.2.2). 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 38.2.1–2.

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 38.2.1–2). 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 38.2.1–2). 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," Section 1.2.2. 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," Section 35.1.5, for details.
- Modified "hard" contact, softened contact, and viscous damping. See "Contact pressure-overclosure relationships," Section 35.1.2, and "Contact damping," Section 35.1.3, 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," Section 35.2.1.

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," Section 34.3.4, for more details on modeling interference fit problems.

Input File Usage: *CONTACT INTERFERENCE, TYPE=ELEMENT

38.2.2 GAP ELEMENT LIBRARY

Product: Abaqus/Standard

References

- "Gap contact elements," Section 38.2.1
- *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 degree of freedom

11

Additional solution variables

None.

Nodal coordinates required

For DGAP elements, and for GAPUNI and GAPUNIT if you specify the contact direction $\mathbf{n}$, 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.

38.3 Tube-to-tube contact elements

- "Tube-to-tube contact elements," Section 38.3.1
- "Tube-to-tube contact element library," Section 38.3.2

38.3.1 TUBE-TO-TUBE CONTACT ELEMENTS

Product: Abaqus/Standard

References

- "Tube-to-tube contact element library," Section 38.3.2
- *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.

Chapter 34, "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. It is not possible to use the surface-based contact approach for problems where two three-dimensional tubes contact each other.

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

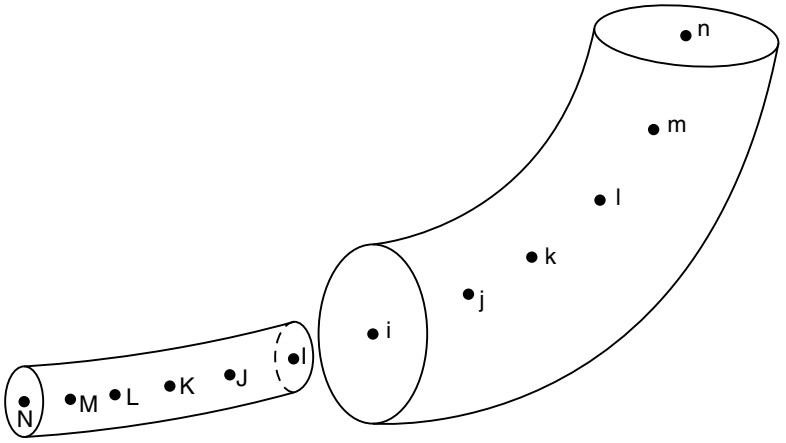
In the case of internal tube-to-tube contact, the slide line can be placed on the inner tube or the outer tube. Generally the slide line should be associated with the outer tube (see Figure 38.3.1–1); however, if the inner tube is stiffer than the outer tube, the slide line should be attached to the inner tube.

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.



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 38.3.1–1 Internal tube-to-tube contact example.

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," Section 36.1.1.

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," Section 35.1.5, for details.
- Modified "hard" contact, softened contact, and viscous damping. See "Contact pressure-overclosure relationships," Section 35.1.2, and "Contact damping," Section 35.1.3, for details.

38.3.2 TUBE-TO-TUBE CONTACT ELEMENT LIBRARY

Product: Abaqus/Standard

References

- "Tube-to-tube contact elements," Section 38.3.1
- *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

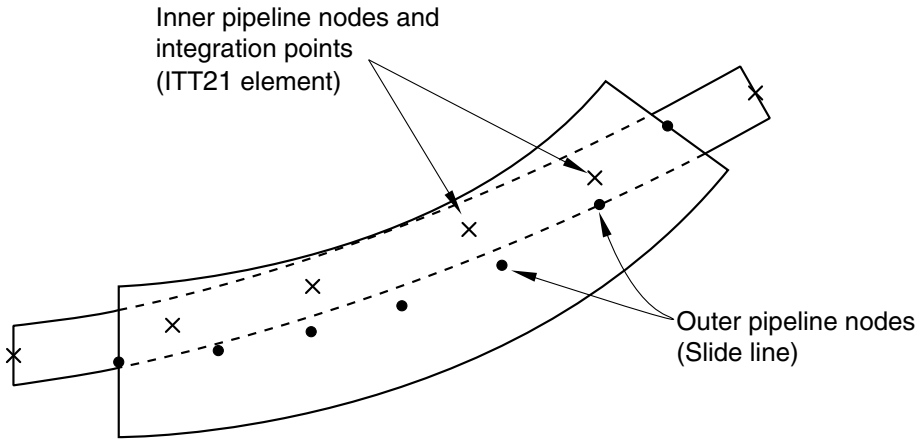
S11	Normal component of the force between the two pipes.
S12	Shear force between the two pipes, parallel to the axis of the second (outer) pipe.
S13	Shear force between the two pipes, normal to the contact direction and to the axis of the second (outer) pipe (for ITT31 only).

Strain components

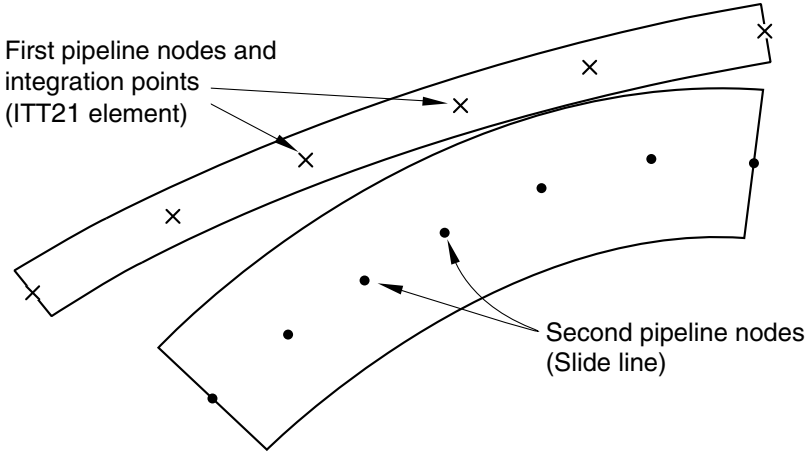
E11	Overclosure of the surfaces in the direction normal to the tangent to the centerline of the second (outer) pipe.
E12	Accumulated relative tangential motion between the two pipes, parallel to the axis of the second (outer) pipe.
E13	Accumulated relative tangential motion between the two pipes, normal to the contact direction and to the axis of the second (outer) pipe (for ITT31 only).

Node ordering and integration point numbering

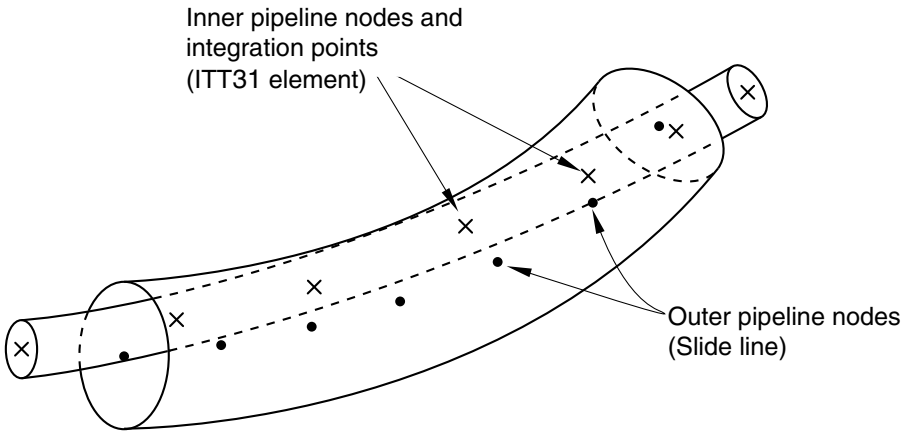
2-D internal tube contact



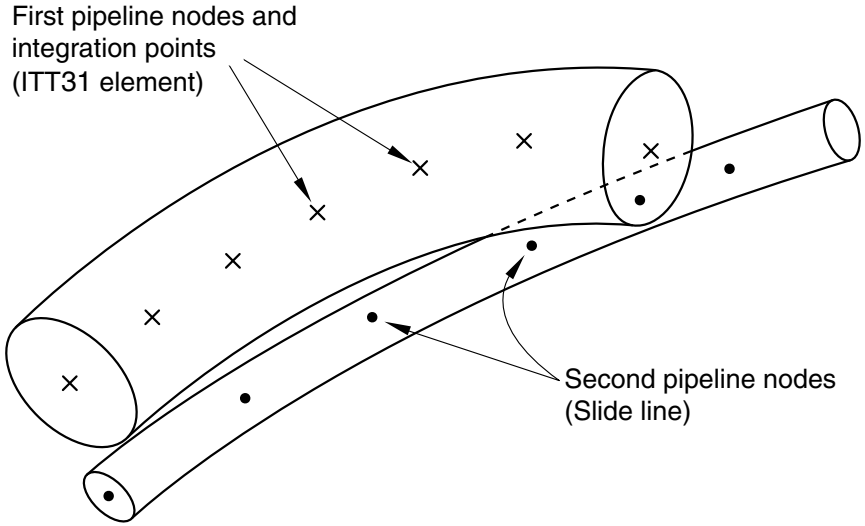
2-D external tube contact



3-D internal tube contact



3-D external tube contact



38.4 Slide line contact elements

- “Slide line contact elements,” Section 38.4.1
- “Axisymmetric slide line element library,” Section 38.4.2

38.4.1 SLIDE LINE CONTACT ELEMENTS

Product: Abaqus/Standard

References

- "Axisymmetric slide line element library," Section 38.4.2
- *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 "master-slave" concepts for enforcing contact constraints seen in surface-based contact.

For a general discussion of contact modeling, see Chapter 34, "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 38.4.1–1). 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.

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 slip directions, $\mathbf{t}_1$ and $\mathbf{t}_2$ (see Figure 38.4.1–2).

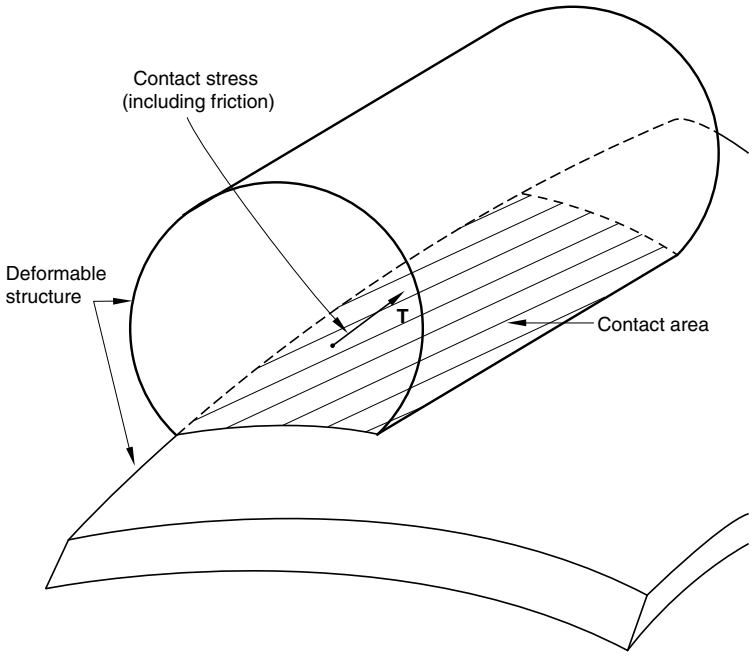


Figure 38.4.1–1 Interaction between deformable structures.

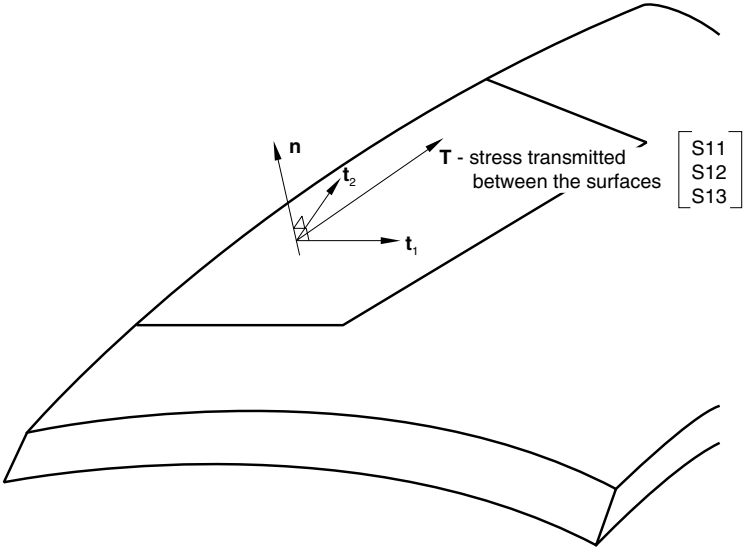


Figure 38.4.1–2 Local system for interface contact normal and shear traction.

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 38.4.1–3), where $\mathbf{s} = (0, 0, 1)$ is the vector that is orthogonal to the contact plane.

As shown in Figure 38.4.1–3, 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.

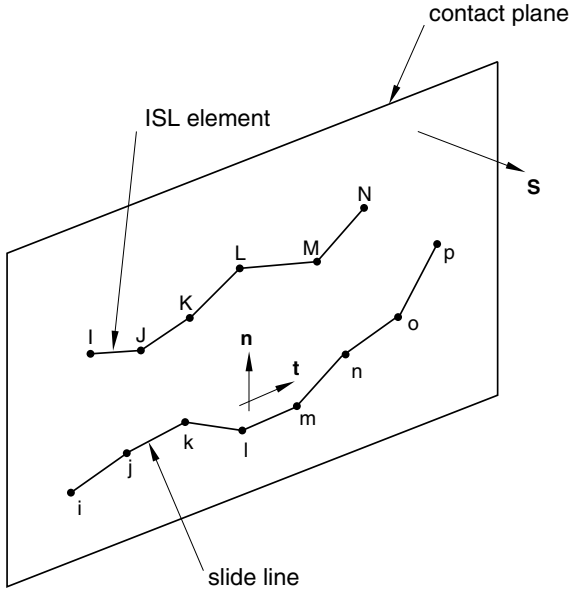


Figure 38.4.1–3 Defining the local basis for a slide line.

The tangent to the slide line coincides with the first slip direction, $\mathbf{t}_1$, of the local basis system. The second slip direction, $\mathbf{t}_2$, is in the opposite direction of $\mathbf{s}$.

The master-slave 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 "master-slave" concept to enforce the contact constraints. The slide line contact elements form the "slave" surface. The nodes that you specify to define the slide line define the "master" surface. The nodes of the slide line contact elements are constrained not to penetrate the master surface.

The considerations for choosing the master and slave surfaces are the same regardless of whether surfaces or elements are used to define contact. The master 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 (master 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 38.4.1–4), 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.

Input File Usage: ∗SLIDE LINE, ELSET=*element_set_name*, TYPE=LINEAR
first node number, second node number, etc.

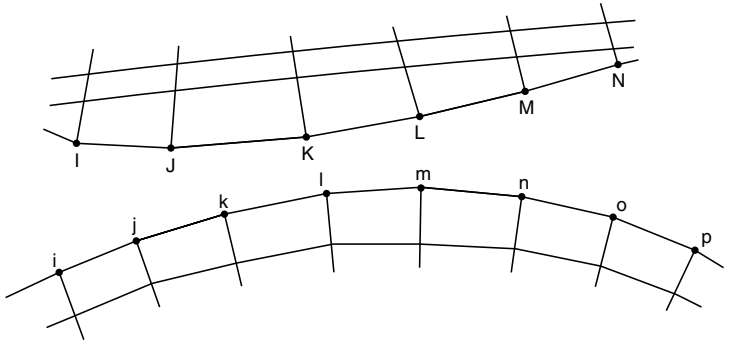


Figure 38.4.1–4 First-order (linear) slide line example.

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 38.4.1–5, 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.

Input File Usage: ∗SLIDE LINE, ELSET=*element_set_name*, TYPE=PARABOLIC
first node number, second node number, etc.

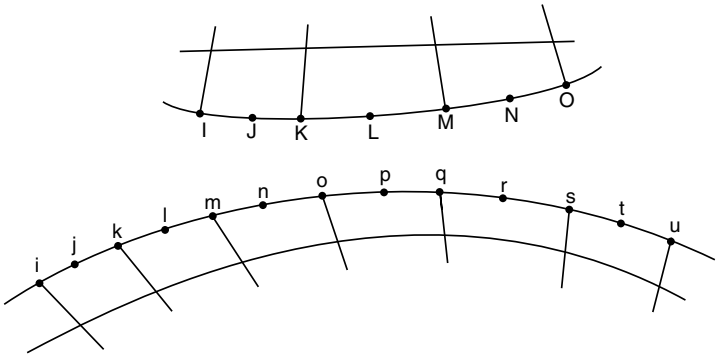


Figure 38.4.1–5 Second-order (parabolic) slide line example.

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," Section 36.1.1.

Defining slide line elements (slave surface)

Many finite-sliding contact simulations can use the surface-based contact approach, described in Chapter 34, "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," Section 34.3.10).

The slide line contact elements define the slave surface. The contact area associated with each node on the slave 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," Section 35.1.5, for details.
- Modified "hard" contact, softened contact, and viscous damping. See "Contact pressure-overclosure relationships," Section 35.1.2, and "Contact damping," Section 35.1.3, 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

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

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 "Defining contact pairs in Abaqus/Standard," Section 34.3.1).

Input File Usage: *TORQUE PRINT, FREQUENCY=*n*

38.4.2 AXISYMMETRIC SLIDE LINE ELEMENT LIBRARY

Product: Abaqus/Standard

References

- "Slide line contact elements," Section 38.4.1
- *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 (master 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

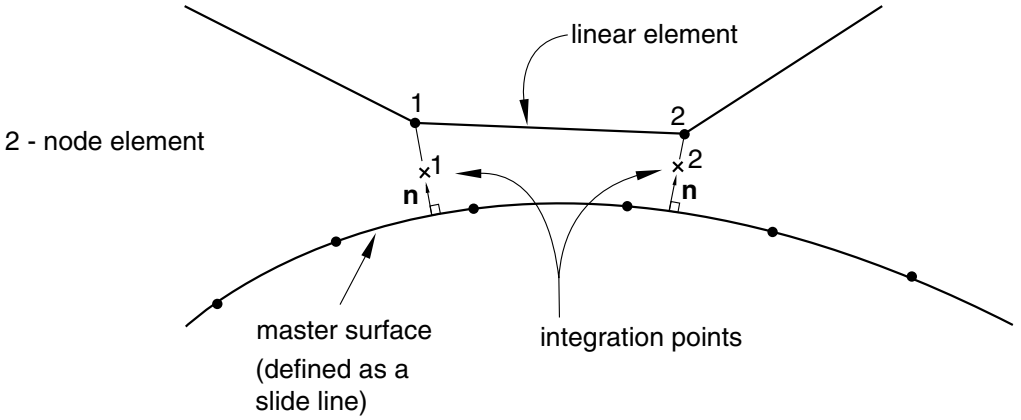
Stress components

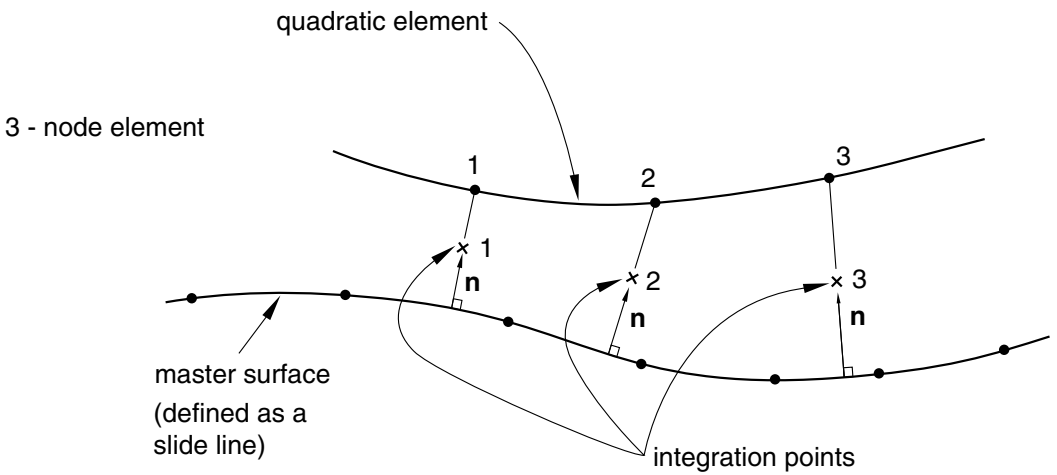
S11	Pressure between the node on the body and the slide line with which it interacts.
S12	Shear stress between the node on the body and the slide line with which it interacts.

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





38.5 Rigid surface contact elements

- “Rigid surface contact elements,” Section 38.5.1
- “Axisymmetric rigid surface contact element library,” Section 38.5.2

38.5.1 RIGID SURFACE CONTACT ELEMENTS

Product: Abaqus/Standard

References

- "Axisymmetric rigid surface contact element library," Section 38.5.2
- "Analytical rigid surface definition," Section 2.3.4
- *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 "master-slave" 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 Chapter 34, "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," Section 34.3.9), or
- when CAXA or SAXA elements are involved in contact (see "Contact modeling if asymmetric-axisymmetric elements are present," Section 34.3.10).

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," Section 2.3.4. 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 master-slave concept for rigid surface contact elements

Rigid surface contact elements use a "master-slave" concept to enforce the contact constraints. The rigid surface contact elements form the "slave" surface, and the nodes of these elements are constrained not to penetrate into the rigid ("master") 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" in "Analytical rigid surface definition," Section 2.3.4.

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 slave 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," Section 35.1.5, for details.
- Modified "hard" contact, softened contact, and viscous damping. See "Contact pressure-overclosure relationships," Section 35.1.2, and "Contact damping," Section 35.1.3, for details.

38.5.2 AXISYMMETRIC RIGID SURFACE CONTACT ELEMENT LIBRARY

Product: Abaqus/Standard

References

- "Analytical rigid surface definition," Section 2.3.4
- "Rigid surface contact elements," Section 38.5.1
- *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.

39. Defining Cavity Radiation in Abaqus/Standard

Defining cavity radiation 39.1

39.1 Defining cavity radiation

- “Cavity radiation,” Section 39.1.1

39.1.1 CAVITY RADIATION

Products: Abaqus/Standard Abaqus/CAE

References

- "Procedures: overview," Section 6.1.1
- "Heat transfer analysis procedures: overview," Section 6.5.1
- *CAVITY DEFINITION
- *COUPLED THERMAL-ELECTRICAL
- *CYCLIC
- *EMISSIVITY
- *HEAT TRANSFER
- *MOTION
- *PERIODIC
- *PHYSICAL CONSTANTS
- *RADIATION FILE
- *RADIATION PRINT
- *RADIATION OUTPUT
- *RADIATION SYMMETRY
- *RADIATION VIEWFACTOR
- *REFLECTION
- *SURFACE
- *SURFACE PROPERTY
- *VIEWFACTOR OUTPUT
- "Cavity radiation," Section 2.11.4 of the Abaqus Theory Manual
- "Defining a cavity radiation interaction," Section 15.13.18 of the Abaqus/CAE User's Manual, in the online HTML version of this manual
- "Defining a cavity radiation interaction property," Section 15.14.3 of the Abaqus/CAE User's Manual, in the online HTML version of this manual

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," Section 6.5.2, and "Coupled thermal-electrical analysis," Section 6.7.3);
- 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," Section 2.11.4 of the Abaqus Theory Manual, and "Procedures: overview," Section 6.1.1). Each cavity defines a viewfactor 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," Section 35.2.1) 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," Section 32.4.4); or
- parallel cavity decomposition for parallel calculation of viewfactors 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," Section 6.5.3). 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.

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 viewfactor definition, which controls the calculation of viewfactors 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 viewfactor calculations in each step (the specifications from the previous step are used if they are not redefined in a step; see "Controlling viewfactor 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 viewfactor matrices (see "Writing the viewfactor 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," Section 2.3.2. 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," Section 2.3.2:

- 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).

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," Section 2.11.4 of the Abaqus Theory Manual). 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 VIEWFACTOR, 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 viewfactor 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 viewfactor calculations; radiation to the external medium will take place only if the deviation of the sum of the viewfactors from unity is more than this tolerance. See "Controlling the accuracy of viewfactor 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 viewfactor matrix and solution of the radiative heat transfer equations (see "Cavity radiation," Section 2.11.4 of the Abaqus Theory Manual). 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 viewfactor 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 viewfactor 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," Section 3.5.2 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," Section 6.5.2); or
- allowing more solver iterations per time increment ("Convergence criteria for nonlinear problems," Section 7.2.3).

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 (master) 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 viewfactor definition. The radiation viewfactor 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 VIEWFACTOR, 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 39.1.1–1. 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 39.1.1–2. 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 39.1.1–3. This type of reflection can be used only with axisymmetric cavities.

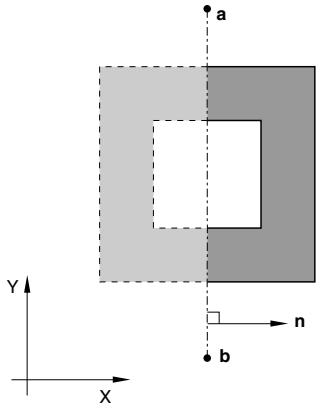


Figure 39.1.1–1 Reflection symmetry through a line.

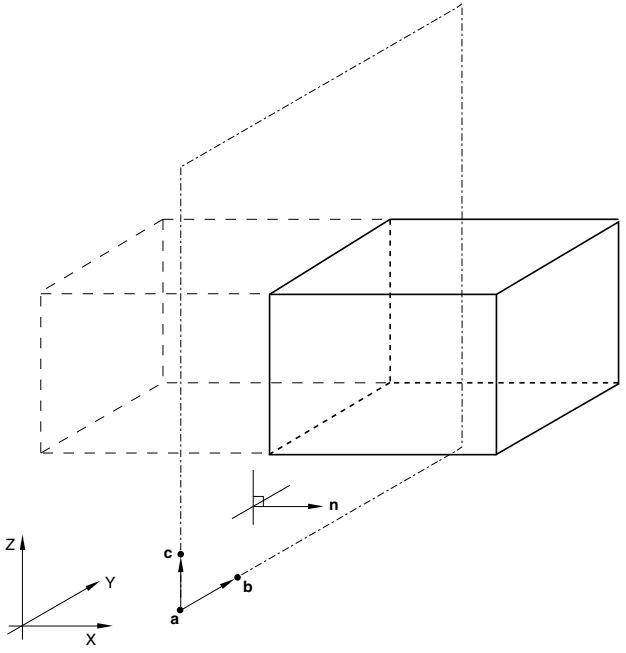


Figure 39.1.1–2 Reflection symmetry through a plane.

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

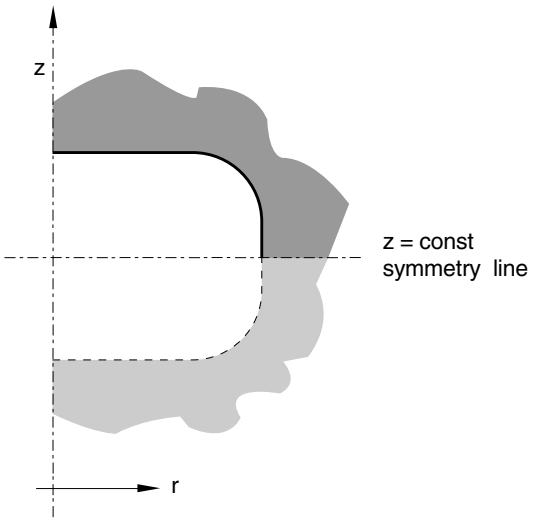


Figure 39.1.1–3 Reflection symmetry through a line of constant *z*-coordinate.

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 viewfactor 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 viewfactor 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 viewfactor calculations" below.) Periodic symmetry for defining the cavity radiation viewfactor 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 39.1.1–4.

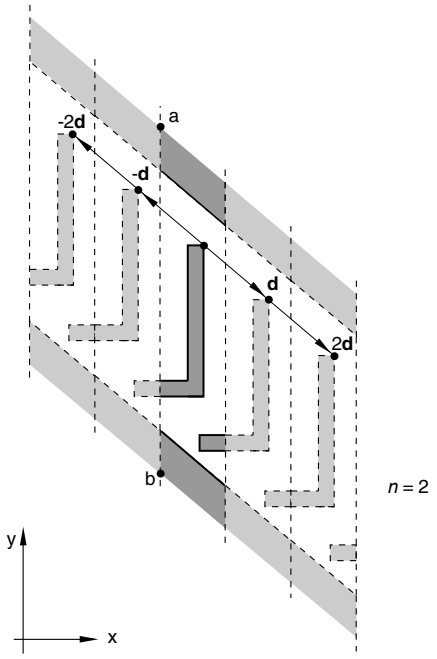


Figure 39.1.1–4 Two-dimensional periodic symmetry.

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 39.1.1–5. 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.

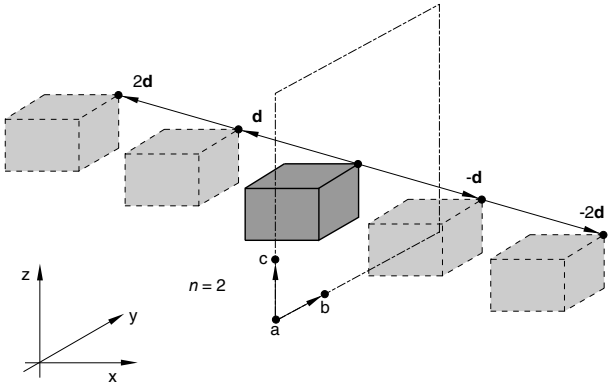


Figure 39.1.1–5 Three-dimensional periodic symmetry.

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 39.1.1–6. 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 39.1.1–7. The cavity surface defined in the model must be bounded by the line *lk* and a line passing through *l* at an

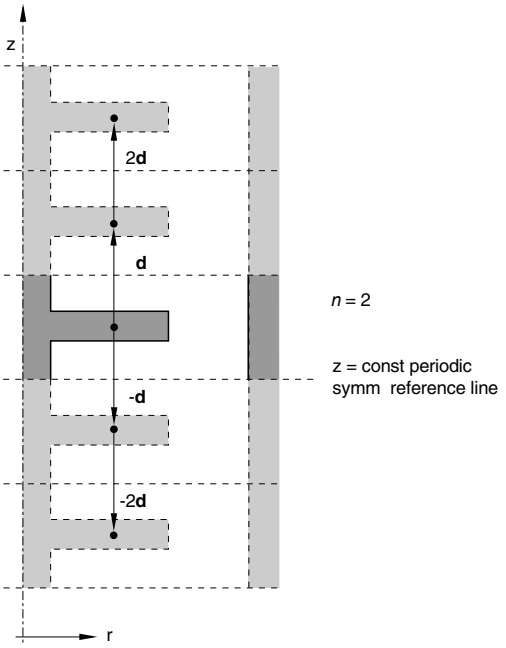


Figure 39.1.1–6 Axisymmetric periodic symmetry.

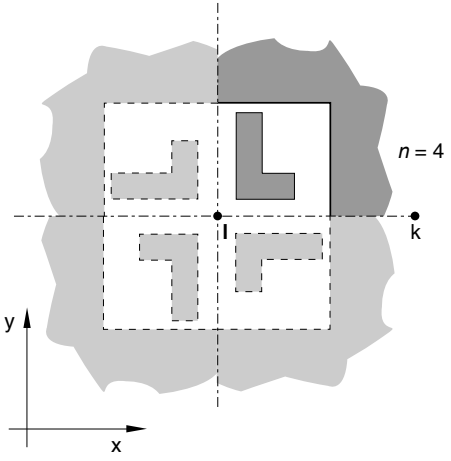


Figure 39.1.1–7 Cyclic symmetry about a point.

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 39.1.1–8. 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.

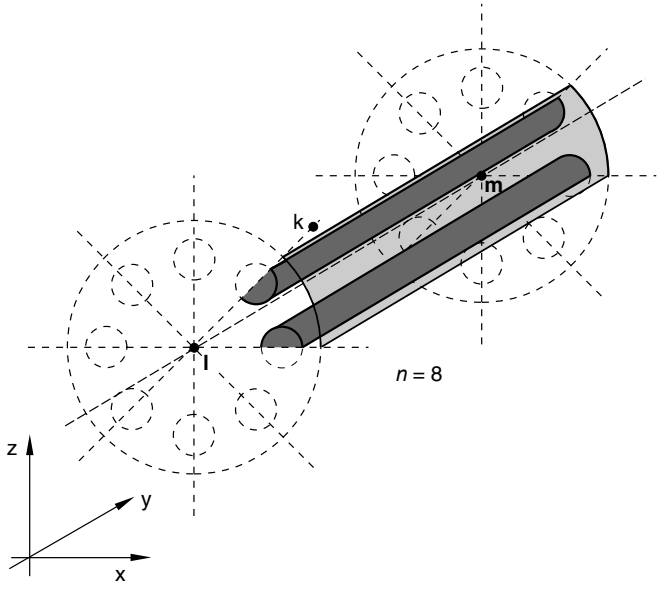


Figure 39.1.1–8 Cyclic symmetry about an axis.

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 39.1.1–1. Figure 39.1.1–9 through Figure 39.1.1–12 illustrate some possible symmetry combinations.

Table 39.1.1–1 Permissible number of symmetry definitions used in combination.

Reflection	Periodic	Cyclic	2-D	3-D	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	•	•		
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.

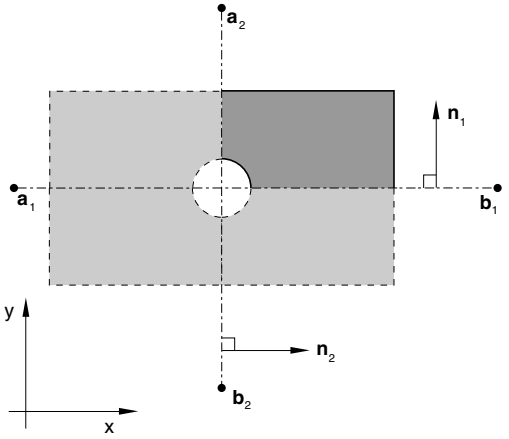


Figure 39.1.1–9 Combination of two reflection symmetries in two dimensions.

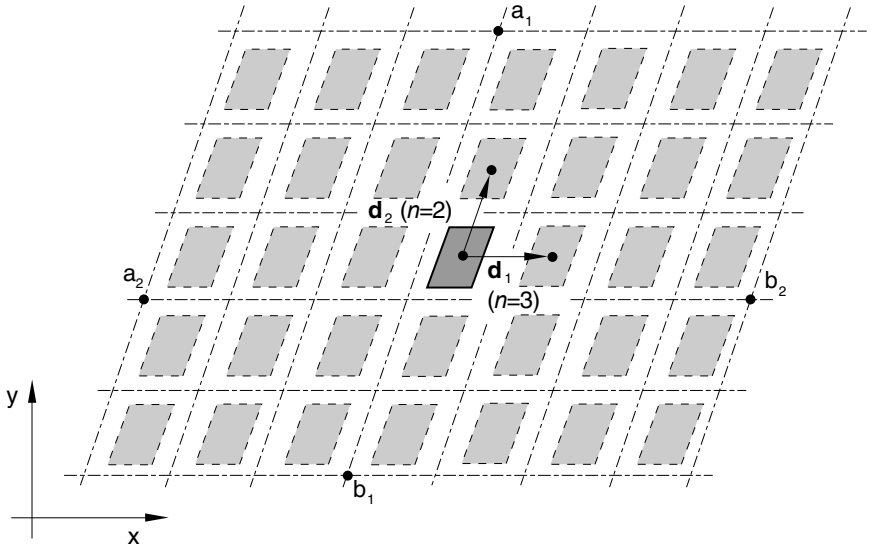


Figure 39.1.1–10 Combination of two periodic symmetries in two dimensions.

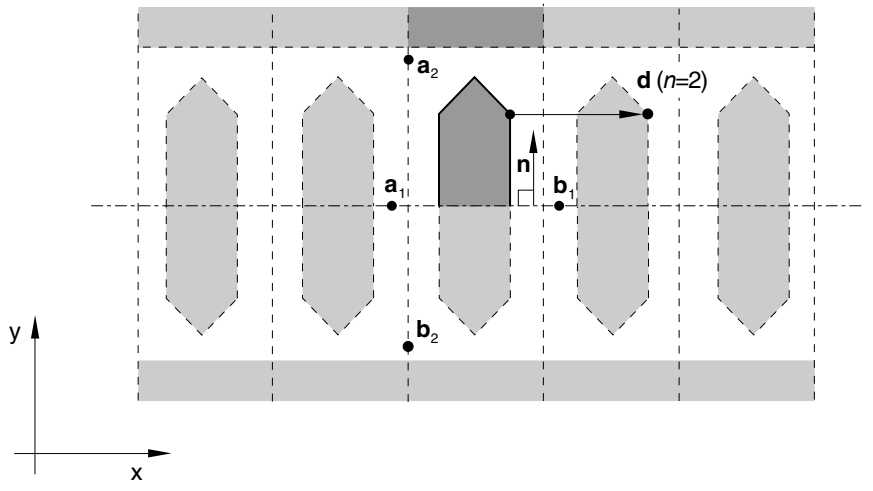


Figure 39.1.1–11 Combination of one reflection symmetry and one periodic symmetry in two dimensions.

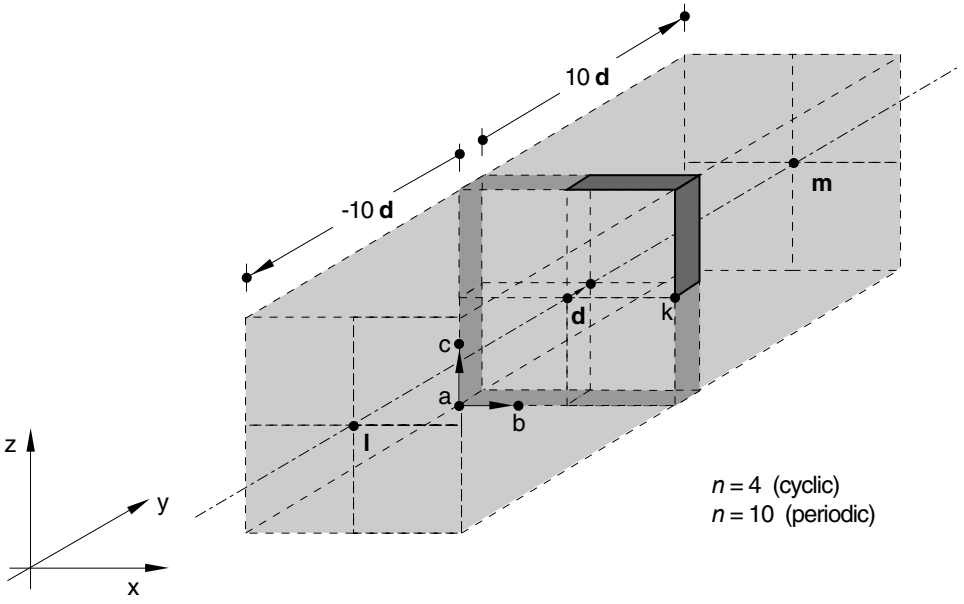


Figure 39.1.1–12 Combination of one cyclic symmetry and one periodic symmetry in three dimensions.

Prescribing motion during a cavity radiation analysis

In many cavity radiation problems such as simulations of manufacturing sequences, radiation viewfactors 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 viewfactors (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," Section 2.1.5).

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 Abaqus/Standard and Abaqus/Explicit," Section 32.3.1) 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," Section 1.2.2.)

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 "Procedures: overview," Section 6.1.1). 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 39.1.1–13.

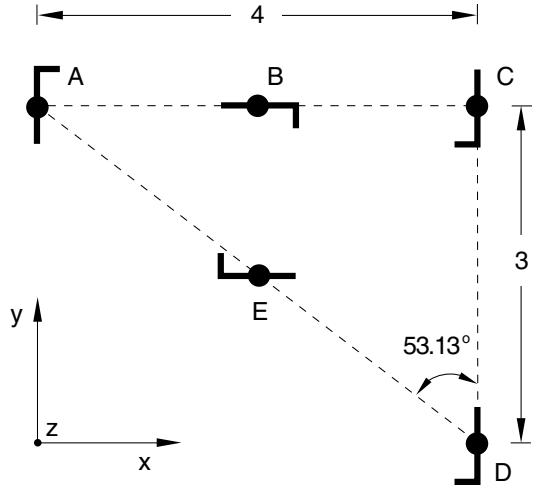


Figure 39.1.1–13 Planar motion example.

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 39.1.1–13. 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," Section 32.1.2).

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 viewfactor recalculation due to motion

You can control how viewfactors 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. Viewfactor 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.

Viewfactor recalculation can be expensive; use discretion when choosing a value for *max*.

Input File Usage: ∗RADIATION VIEWFACTOR, MDISP=*max*, NSET=*nset*

The *max* and *nset* values must always be specified together.

Abaqus/CAE Usage: Viewfactor recalculation due to motion is not supported with cavity radiation in Abaqus/CAE.

Controlling viewfactor calculation during the analysis

The cavity radiation capability can be used in applications such as the simulation of manufacturing sequences where radiation viewfactors change during the simulation. Therefore, radiation viewfactor definitions provide significant flexibility for the control of viewfactor calculations during a step.

Multiple radiation viewfactor definitions can be specified within a step definition if different types of radiation and viewfactor calculations are required for different cavities. Different types of viewfactor calculations can be specified for the same cavity in different steps of the analysis.

By default, viewfactors are calculated at the beginning of the first step that includes a radiation viewfactor definition. Viewfactors are recalculated at the beginning of a subsequent step only if the viewfactor 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 viewfactors from the user-specified restart step and increment and recalculates the viewfactors only if the viewfactor definitions have changed.

You can specify the name of the cavity for which radiation viewfactor control is being specified. If you do not specify a cavity name, the radiation viewfactor definition applies to all cavities in the model.

Input File Usage: *RADIATION VIEWFACTOR, CAVITY=*cavity_name*

Abaqus/CAE Usage: Radiation viewfactors 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 viewfactor 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 viewfactors 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 viewfactors 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 viewfactor calculation off for a step:

*RADIATION VIEWFACTOR, OFF

Use one of the following options to turn viewfactor calculation back on in a subsequent step:

*RADIATION VIEWFACTOR
*RADIATION VIEWFACTOR, MDISP=*max*, NSET=*nset*

Abaqus/CAE Usage: Radiation viewfactors 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 viewfactor calculations

Abaqus/Standard uses a progressive integration scheme for viewfactor 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 viewfactor. See "Viewfactor calculation," Section 2.11.5 of the Abaqus Theory Manual, 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 $\boldsymbol{d}$ 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 viewfactor 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 VIEWFACTOR, LUMPED AREA=*P1*, INFINITESIMAL=*P2*, INTEGRATION=*integration points per edge*

Abaqus/CAE Usage: Interaction module: **Create Interaction**: **Cavity radiation**: **Viewfactors**: enter new values or accept the defaults for **Infinitesimal facet area ratio**, **Gauss integration points per edge**, and **Lumped area distance-square value**

Viewfactor calculation checks for closed cavities

You can provide a tolerance on the accuracy of the viewfactor calculation. In a closed cavity the sum of the viewfactors for each cavity facet should be one. Abaqus/Standard compares the value of the specified tolerance to the deviation of the average sum from unity (the average sum is computed by summing the sums of the viewfactors over all the facets and dividing it by the number of facets). If the tolerance is violated for a closed cavity, the analysis is terminated. The default viewfactor tolerance is 0.05. Failure to meet this criterion may indicate a need for mesh refinement.

Input File Usage: *RADIATION VIEWFACTOR, VTOL=*tolerance*

Abaqus/CAE Usage: Interaction module: **Create interaction**: **Cavity radiation**: **Viewfactors**: **Accuracy tolerance**: *tolerance*

Viewfactor calculations in cavities with symmetries

The viewfactor calculations account for the closure of a cavity implied by any cavity symmetries. For cavities without periodic or cyclic symmetries the viewfactors 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," Section 35.2.1).

Viewfactor calculations in open cavities

If the sum of the viewfactors for facets in an open cavity (defined by specifying a value for the ambient temperature) deviates from unity by more than the specified viewfactor tolerance, radiation to the ambience will take place. 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 loosen the viewfactor 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 39.1.1–14); "blocking" may occur in geometrically complex cavities.

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.

Viewfactor calculations with blocking surfaces are especially sensitive to mesh refinement. If a mesh is too coarse, the viewfactors may not add up to one (in a closed cavity). To obtain accurate results, the mesh should be refined until the viewfactors 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 VIEWFACTOR, 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 VIEWFACTOR, BLOCKING=PARTIAL

Abaqus/CAE Usage: Interaction module: **Create interaction**: **Cavity radiation**: **Properties**: **Blocking surface checks**: **Partial**

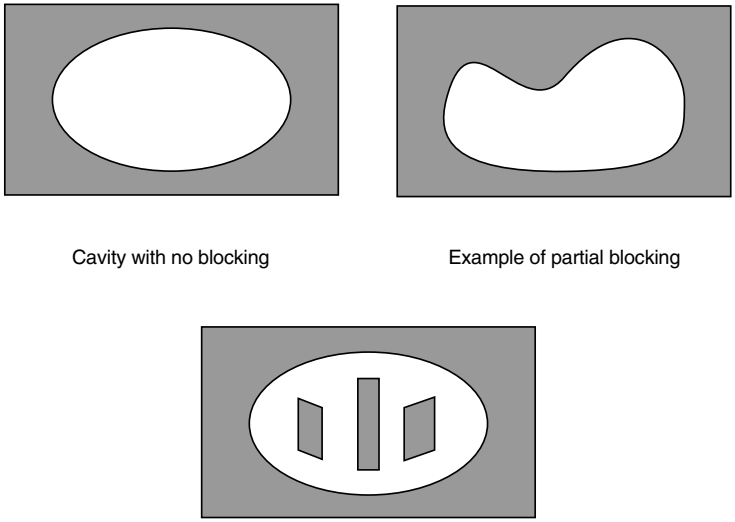


Figure 39.1.1–14 Illustrations of blocking.

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 VIEWFACTOR, 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 viewfactors need not be calculated, which reduces the computational effort required for the viewfactor calculations.

Input File Usage: *RADIATION VIEWFACTOR, RANGE=*distance*

Abaqus/CAE Usage: Interaction module: **Create interaction**: **Cavity radiation**: **Viewfactors**: 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 master surface in a tied coupling constraint ("Mesh tie constraints," Section 33.3.1); the multiple-degree-of-freedom, physical, heat transfer shell element forms the slave 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 "Initial conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.2.1.

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," Section 6.5.2).

Boundary conditions

You can specify boundary conditions to prescribe temperatures (degree of freedom 11) at the nodes (see "Boundary conditions in Abaqus/Standard and Abaqus/Explicit," Section 32.3.1). Shell elements have additional temperature degrees of freedom 12, 13, etc. through the thickness (see "Conventions," Section 1.2.2). Boundary conditions can be specified as functions of time by referring to amplitude curves ("Amplitude curves," Section 32.1.2).

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," Section 32.4.4:

- 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," Section 32.6.1.

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," Section 6.5.2, and "Thermal properties: overview," Section 25.2.1.

You can specify internal heat generation—see "Internal heat generation" in "Uncoupled heat transfer analysis," Section 6.5.2.

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," Section 26.1.3; "Uncoupled heat transfer analysis," Section 6.5.2; and "Coupled thermal-electrical analysis," Section 6.7.3). 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 viewfactor for a facet (sum of the viewfactor values in the row of the viewfactor matrix corresponding to the facet).
FTEMP	Facet temperature.

All of the output variables are listed in "Abaqus/Standard output variable identifiers," Section 4.2.1. Abaqus/CAE supports motion display and can display surface- and element-based results.

Writing the viewfactor matrices to the results file

You can write the viewfactor 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 viewfactor matrix is written for each cavity radiation element in the specified cavity.

You can control the frequency of viewfactor 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 output format," Section 5.1.2. The file can be written in binary or ASCII format (see "Controlling the format of the results file in Abaqus/Standard" in "Output," Section 4.1.1).

Input File Usage: *VIEWFACTOR OUTPUT, CAVITY=*cavity_name*, FREQUENCY=*n*

Abaqus/CAE Usage: Viewfactor 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, viewfactor 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=Δε_max, DELTMX=Δθ_max
Data line to define incrementation
*RADIATION VIEWFACTOR, 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 VIEWFACTOR, OFF
*CFLUX, OP=NEW
*DFLUX, OP=NEW
*END STEP
```

About SIMULIA

SIMULIA is the Dassault Systèmes brand that delivers a scalable portfolio of Realistic Simulation solutions including the Abaqus product suite for Unified Finite Element Analysis; multiphysics solutions for insight into challenging engineering problems; and lifecycle management solutions for managing simulation data, processes, and intellectual property. By building on established technology, respected quality, and superior customer service, SIMULIA makes realistic simulation an integral business practice that improves product performance, reduces physical prototypes, and drives innovation. Headquartered in Providence, RI, USA, with R&D centers in Providence and in Vélizy, France, SIMULIA provides sales, services, and support through a global network of regional offices and distributors. For more information, visit www.simulia.com.

About Dassault Systèmes

As a world leader in 3D and Product Lifecycle Management (PLM) solutions, Dassault Systèmes brings value to more than 100,000 customers in 80 countries. A pioneer in the 3D software market since 1981, Dassault Systèmes develops and markets PLM application software and services that support industrial processes and provide a 3D vision of the entire lifecycle of products from conception to maintenance to recycling. The Dassault Systèmes portfolio consists of CATIA for designing the virtual product, SolidWorks for 3D mechanical design, DELMIA for virtual production, SIMULIA for virtual testing, ENOVIA for global collaborative lifecycle management, and 3DVIA for online 3D lifelike experiences. Dassault Systèmes' shares are listed on Euronext Paris (#13065, DSY.PA), and Dassault Systèmes' ADRs may be traded on the US Over-The-Counter (OTC) market (DASTY). For more information, visit www.3ds.com.

www.3ds.com Abaqus, the 3DS logo, SIMULIA, CATIA, SolidWorks, DELMIA, ENOVIA, 3DVIA, and Unified FEA are trademarks or registered trademarks of Dassault Systèmes or its subsidiaries in the US and/or other countries. Other company, product, and service names may be trademarks or service marks of their respective owners. © Dassault Systèmes, 2011

DASSAULT SYSTEMES