

# COMSOL MULTIPHYSICS®

MODELING  
GUIDE

**VERSION 3.5**



**How to contact COMSOL:****Benelux**

COMSOL BV  
Röntgenlaan 19  
2719 DX Zoetermeer  
The Netherlands  
Phone: +31 (0) 79 363 4230  
Fax: +31 (0) 79 361 4212  
info@comsol.nl  
www.comsol.nl

**Denmark**

COMSOL A/S  
Diplomvej 376  
2800 Kgs. Lyngby  
Phone: +45 88 70 82 00  
Fax: +45 88 70 80 90  
info@comsol.dk  
www.comsol.dk

**Finland**

COMSOL OY  
Arabianranta 6  
FIN-00560 Helsinki  
Phone: +358 9 2510 400  
Fax: +358 9 2510 4010  
info@comsol.fi  
www.comsol.fi

**France**

COMSOL France  
WTC, 5 pl. Robert Schuman  
F-38000 Grenoble  
Phone: +33 (0)4 76 46 49 01  
Fax: +33 (0)4 76 46 07 42  
info@comsol.fr  
www.comsol.fr

**Germany**

COMSOL Multiphysics GmbH  
Berliner Str. 4  
D-37073 Göttingen  
Phone: +49-551-99721-0  
Fax: +49-551-99721-29  
info@comsol.de  
www.comsol.de

**Italy**

COMSOL S.r.l.  
Via Vittorio Emanuele II, 22  
25122 Brescia  
Phone: +39-030-3793800  
Fax: +39-030-3793899  
info.it@comsol.com  
www.it.comsol.com

**Norway**

COMSOL AS  
Søndre gate 7  
NO-7485 Trondheim  
Phone: +47 73 84 24 00  
Fax: +47 73 84 24 01  
info@comsol.no  
www.comsol.no

**Sweden**

COMSOL AB  
Tegnérsgatan 23  
SE-111 40 Stockholm  
Phone: +46 8 412 95 00  
Fax: +46 8 412 95 10  
info@comsol.se  
www.comsol.se

**Switzerland**

FEMLAB GmbH  
Technoparkstrasse 1  
CH-8005 Zürich  
Phone: +41 (0)44 445 2140  
Fax: +41 (0)44 445 2141  
info@femlab.ch  
www.femlab.ch

**United Kingdom**

COMSOL Ltd.  
UH Innovation Centre  
College Lane  
Hatfield  
Hertfordshire AL10 9AB  
Phone: +44-(0)-1707 636020  
Fax: +44-(0)-1707 284746  
info.uk@comsol.com  
www.uk.comsol.com

**United States**

COMSOL, Inc.  
1 New England Executive Park  
Suite 350  
Burlington, MA 01803  
Phone: +1-781-273-3322  
Fax: +1-781-273-6603

COMSOL, Inc.  
10850 Wilshire Boulevard  
Suite 800  
Los Angeles, CA 90024  
Phone: +1-310-441-4800  
Fax: +1-310-441-0868

COMSOL, Inc.  
744 Cowper Street  
Palo Alto, CA 94301  
Phone: +1-650-324-9935  
Fax: +1-650-324-9936

info@comsol.com  
www.comsol.com

For a complete list of international  
representatives, visit  
www.comsol.com/contact

**Company home page**  
www.comsol.com

**COMSOL user forums**  
www.comsol.com/support/forums

*COMSOL Multiphysics Modeling Guide*

© COPYRIGHT 1994–2008 by COMSOL AB. All rights reserved

Patent pending

The software described in this document is furnished under a license agreement. The software may be used or copied only under the terms of the license agreement. No part of this manual may be photocopied or reproduced in any form without prior written consent from COMSOL AB.

COMSOL, COMSOL Multiphysics, COMSOL Script, COMSOL Reaction Engineering Lab, and FEMLAB are registered trademarks of COMSOL AB.

Other product or brand names are trademarks or registered trademarks of their respective holders.

Version: September 2008 COMSOL 3.5

Part number: CM020003

# C O N T E N T S

## Chapter 1: Introduction

<b>Overview of the COMSOL Multiphysics Application Modes</b>	<b>2</b>
Application Modes in COMSOL Multiphysics . . . . .	2
Selecting an Application Mode . . . . .	5
<b>Modeling Guidelines</b>	<b>7</b>
Using Symmetries . . . . .	7
Effective Memory Management . . . . .	8
Selecting an Element Type . . . . .	9
Analyzing Model Convergence and Accuracy . . . . .	9
Achieving Convergence When Solving Nonlinear Equations. . . . .	10
Avoiding Strong Transients . . . . .	11
Typographical Conventions . . . . .	11

## Chapter 2: Using the Physics Modes

<b>The Physics Modes</b>	<b>14</b>
Defining the Physics for a Model . . . . .	14
The Physics Modes . . . . .	15
Physics Mode Documentation . . . . .	16

## Chapter 3: Acoustics

<b>Fundamentals of Acoustics</b>	<b>20</b>
What is Acoustics? . . . . .	20
Five Standard Acoustics Problems . . . . .	20
Mathematical Models for Acoustic Analysis . . . . .	21
<b>The Acoustics Application Mode</b>	<b>23</b>
Variables and Space Dimensions . . . . .	23

PDE Formulation . . . . .	23
Subdomain Settings . . . . .	24
Boundary Conditions . . . . .	25
Scalar Variables . . . . .	28
Application Mode Variables . . . . .	29
Units . . . . .	31
2D . . . . .	31
2D Axisymmetric. . . . .	33
<b>Example—Reactive Muffler</b>	<b>35</b>
Introduction . . . . .	35
Model Definition . . . . .	35
Results and Discussion. . . . .	37
Reference . . . . .	38
Modeling Using the Graphical User Interface . . . . .	38

## Chapter 4: Diffusion

<b>The Diffusion Application Mode</b>	<b>46</b>
PDE Formulation . . . . .	46
Subdomain Settings . . . . .	46
Boundary Conditions . . . . .	47
Application Mode Variables . . . . .	49
<b>The Convection and Diffusion Application Mode</b>	<b>51</b>
PDE Formulation . . . . .	51
Subdomain Settings . . . . .	52
Boundary Conditions . . . . .	54
Application Mode Variables . . . . .	56
References . . . . .	57
<b>Example—Effective Diffusivity in Porous Materials</b>	<b>58</b>
Introduction . . . . .	58
Model Definition . . . . .	59
Results and Discussion. . . . .	60
Modeling in COMSOL Multiphysics . . . . .	63



Modeling Using the Graphical User Interface . . . . .	64
---	----

## Chapter 5: Electromagnetics

<b>The Electromagnetics Application Modes</b>	<b>78</b>
<b>Fundamentals of Electromagnetics</b>	<b>79</b>
Electromagnetic Force for Particle Tracing . . . . .	83
Electromagnetic Forces . . . . .	84
<b>The Conductive Media DC Application Mode</b>	<b>85</b>
PDE Formulation . . . . .	85
Boundary Conditions . . . . .	86
Line Sources . . . . .	89
Point Sources . . . . .	89
Application Mode Variables . . . . .	89
<b>The Electrostatics Application Mode</b>	<b>91</b>
PDE Formulation . . . . .	91
Application Scalar Variables . . . . .	92
Boundary Conditions . . . . .	92
Line Sources . . . . .	94
Point Sources . . . . .	94
Application Mode Variables . . . . .	94
<b>Magnetostatics Application Mode</b>	<b>97</b>
PDE Formulation . . . . .	97
Application Mode Property . . . . .	98
Application Scalar Variable . . . . .	98
Boundary and Interface Conditions . . . . .	98
Point Sources . . . . .	100
Application Mode Variables . . . . .	100
<b>AC Power Electromagnetics</b>	<b>107</b>
PDE Formulation . . . . .	107
Application Scalar Variables . . . . .	108

Boundary and Interface Conditions . . . . .	108
Point Sources . . . . .	110
Application Mode Variables . . . . .	110
<b>Example—Electric Sensor</b>	<b>121</b>
Model Definition . . . . .	121
Results and Discussion. . . . .	121
Modeling Using the Graphical User Interface . . . . .	122
<b>Example—A Permanent Magnet</b>	<b>126</b>
Modeling Using the Graphical User Interface . . . . .	126
Alternative Modeling Approach . . . . .	130
References . . . . .	131

## Chapter 6: Fluid Mechanics

<b>The Navier-Stokes Application Mode</b>	<b>134</b>
Variables and Space Dimension . . . . .	134
PDE Formulation and Equations . . . . .	134
Subdomain Settings . . . . .	135
Boundary Conditions . . . . .	136
Point Settings . . . . .	144
Numerical Stability—Stabilization Techniques . . . . .	144
Corner Smoothing . . . . .	147
Application Mode Variables . . . . .	148
Solver Settings . . . . .	149
Khan and Richardson Force for Particle Tracing . . . . .	152
References . . . . .	153
<b>Example—Steady Incompressible Flow</b>	<b>154</b>
Introduction . . . . .	154
Model Definition . . . . .	154
Results. . . . .	155
References . . . . .	158
Modeling Using the Graphical User Interface . . . . .	159

# Chapter 7: Heat Transfer

<b>Heat Transfer Fundamentals</b>	<b>168</b>
What Is Heat Transfer? . . . . .	168
The Heat Equation . . . . .	169
<b>The Conduction Application Mode</b>	<b>171</b>
Variables and Space Dimensions . . . . .	171
PDE Formulation . . . . .	171
Subdomain Settings . . . . .	172
Boundary Condition Types . . . . .	173
Boundary Settings . . . . .	176
Application Mode Variables . . . . .	177
<b>The Convection and Conduction Application Mode</b>	<b>179</b>
Variables and Space Dimensions . . . . .	179
PDE Formulation . . . . .	179
Subdomain Settings . . . . .	180
Boundary Conditions . . . . .	184
Application Mode Variables . . . . .	185
References . . . . .	186
<b>Examples of Heat Transfer Models</b>	<b>188</b>
1D Steady-State Heat Transfer with Radiation . . . . .	188
Model Definition . . . . .	188
Results. . . . .	189
Modeling Using the Graphical User Interface . . . . .	189
2D Steady-State Heat Transfer with Convection . . . . .	192
Model Definition . . . . .	192
Results. . . . .	193
Modeling Using the Graphical User Interface . . . . .	194
2D Axisymmetric Transient Heat Transfer . . . . .	196
Model Definition . . . . .	196
Results. . . . .	197
Modeling Using the Graphical User Interface . . . . .	198
References . . . . .	202

# Chapter 8: Structural Mechanics

<b>The Structural Mechanics Application Modes</b>	<b>204</b>
<b>Theory Background</b>	<b>205</b>
Strain-Displacement Relationship . . . . .	205
Stress-Strain Relationship . . . . .	205
Implementation . . . . .	207
<b>Application Mode Descriptions</b>	<b>210</b>
Application Mode Properties . . . . .	210
Material Properties . . . . .	212
Constraints . . . . .	213
Loads . . . . .	214
Damping . . . . .	215
<b>The Solid, Stress-Strain Application Mode</b>	<b>216</b>
Variables and Space Dimensions . . . . .	216
PDE Formulation . . . . .	216
Application Mode Variables . . . . .	217
<b>The Plane Stress Application Mode</b>	<b>220</b>
Material . . . . .	220
PDE Formulation . . . . .	220
Application Mode Variables . . . . .	221
<b>The Plane Strain Application Mode</b>	<b>225</b>
Material Properties . . . . .	225
PDE Formulation . . . . .	226
Application Mode Variables . . . . .	226
<b>The Axial Symmetry, Stress-Strain Application Mode</b>	<b>230</b>
PDE Formulation . . . . .	231
Application Mode Variables . . . . .	232
<b>Examples of Structural Mechanics Models</b>	<b>235</b>
Tapered Membrane End Load . . . . .	235

Modeling in COMSOL Multiphysics . . . . .	236
Results. . . . .	236
Modeling Using the Graphical User Interface . . . . .	237
Tapered Cantilever Gravity Load . . . . .	240
Modeling in COMSOL Multiphysics . . . . .	240
Results. . . . .	241
Modeling Using the Graphical User Interface . . . . .	241
Reference . . . . .	244

## Chapter 9: PDE Modes for Equation-Based Modeling

<b>The PDE Modes</b>	<b>246</b>
<b>Using the PDE Modes</b>	<b>247</b>
Starting a Model Using a PDE Mode. . . . .	247
Using the Coefficient Form PDEs. . . . .	247
Using the General Form PDE . . . . .	257
Solving Time-Dependent Problems . . . . .	264
Solving Eigenvalue Problems. . . . .	266
Interpreting PDE Coefficients . . . . .	267
Classical PDEs . . . . .	268
Equation Variables . . . . .	268
Boundary-Coupled Equation Variables. . . . .	270
Ideal and Non-Ideal Constraints . . . . .	273
Variables for PDEs in Weak Form . . . . .	274
Application Mode Variables . . . . .	277
PDE Coefficients and Boundary Conditions with Time Derivatives . . . . .	279
<b>Using Equation Contributions on Interior Mesh Boundaries</b>	<b>280</b>
Specifying Equation Contributions . . . . .	280
Specifying Interior Mesh Boundary Expressions. . . . .	281
<b>Examples—Equation-Based Modeling</b>	<b>282</b>
Poisson’s Equation on the Unit Disk. . . . .	282
Results. . . . .	283
Modeling Using the Graphical User Interface . . . . .	283

Implementing a Point Source . . . . .	285
Modeling Using the Graphical User Interface . . . . .	287

## Chapter 10: Sensitivity Analysis

<b>The Sensitivity Analysis Application Mode</b>	<b>292</b>
Introduction . . . . .	292
Sensitivity Variable Selection . . . . .	292
Scalar Objective Functions . . . . .	294
Model Navigator . . . . .	298
Application Mode Properties . . . . .	298
Subdomain, Boundary, and Edge Settings . . . . .	298
Point Settings . . . . .	300
Scalar Settings . . . . .	301
<b>Example—Predicting the Effect of a Geometrical Change</b>	<b>303</b>
Introduction . . . . .	303
Model Definition . . . . .	303
Results and Discussion. . . . .	306
Modeling Using the Graphical User Interface . . . . .	309

## Chapter 11: Optimization

<b>The Optimization Application Mode</b>	<b>314</b>
Introduction . . . . .	314
Optimization Problem Formulation . . . . .	314
Model Navigator . . . . .	316
Application Mode Properties . . . . .	317
Subdomain Settings . . . . .	317
Boundary Settings . . . . .	321
Edge Settings (3D Only) . . . . .	322
Point Settings . . . . .	322
Scalar Settings . . . . .	323
Solver Settings . . . . .	325

<b>Example—Minimizing the Flow Velocity in a Microchannel</b>	<b>330</b>
Introduction . . . . .	330
Model Definition . . . . .	330
Results and Discussion. . . . .	331
Reference . . . . .	334
Modeling in COMSOL Multiphysics . . . . .	334
Modeling Using the Graphical User Interface . . . . .	335
Computing the Solution—Static Flow . . . . .	338

## Chapter 12: Global Equations and ODEs

<b>Solving ODEs and Global Equations</b>	<b>344</b>
Adding ODEs and Other Global Equations . . . . .	344
The Global Equations Dialog Box. . . . .	344
Solving ODEs—the Lotka-Volterra Equations . . . . .	345
Solving Algebraic and Transcendental Equations . . . . .	346
Adding an ODE to a Boundary . . . . .	347

## Chapter 13: The Weak Form

<b>Weak Form Modeling</b>	<b>351</b>
Adding Weak Form Contributions . . . . .	351
Modeling with PDEs on Boundaries, Edges, and Points. . . . .	352
Using Weak Coefficients or the Weak Modes . . . . .	352
<b>Example—Variational Calculus</b>	<b>354</b>
Model Definition . . . . .	354
Results. . . . .	355
Modeling in COMSOL Multiphysics . . . . .	355
Modeling Using the Graphical User Interface . . . . .	356
Modeling Using the Programming Language . . . . .	357
<b>Using Weak Constraints</b>	<b>358</b>
Specifying Weak Constraints . . . . .	359
Weak Constraints in Assemblies . . . . .	360

Limitations of Weak Constraints . . . . .	361
<b>Example—Coupling Variables and Boundary Constraints</b>	<b>362</b>
Boundary Constraints in a Heat Transfer Model . . . . .	362
No Weak Constraints . . . . .	366
Weak Ideal Constraints . . . . .	366
Weak Non-Ideal Constraints . . . . .	366
<b>Theoretical Background</b>	<b>368</b>
Deriving the Weak Form . . . . .	368
Weak Form Application Modes . . . . .	369
Entering a PDE in Coefficient Form Using the Weak Form . . . . .	373

## Chapter 14: Multiphysics Modeling

<b>Creating Multiphysics Models</b>	<b>376</b>
Multiphysics Modeling Approaches . . . . .	376
Using Predefined Multiphysics Couplings . . . . .	376
Adding Physics Sequentially . . . . .	386
Building a Coupled Multiphysics Model Directly . . . . .	387
Removing Application Modes . . . . .	387
Multiphysics Model Properties . . . . .	388
Managing the Solver and Solution Components . . . . .	388
Deactivating Physics or Equations . . . . .	391
Using the Model Navigator for Multiphysics Models . . . . .	392
Specifying Settings for Multiphysics Models . . . . .	393
<b>Example—Resistive Heating</b>	<b>395</b>
Introduction . . . . .	395
Model Definition . . . . .	395
Modeling in COMSOL Multiphysics . . . . .	396
Modeling Using the Graphical User Interface . . . . .	396
<b>Working with Components</b>	<b>409</b>
The Component Library . . . . .	409
Adding Components . . . . .	410



Merging Components . . . . .	410
------------------------------	-----

## Chapter 15: Using Assemblies

<b>Creating Assemblies</b>	<b>414</b>
The Analyzed Geometry—Assembly vs. Composite Geometry . . . . .	414
Creating Pairs . . . . .	415
<b>Identity Pairs</b>	<b>418</b>
The Identity Pair Dialog Box . . . . .	419
Selection Colors for Pairs . . . . .	422
<b>Specifying Physics Settings on Pairs</b>	<b>423</b>
Controlling the Use of Pairs. . . . .	424
Edge and Point Pairs . . . . .	425
Slit Boundary Conditions. . . . .	425
<b>Meshing Assemblies</b>	<b>427</b>
Example—Meshing a Composite Geometry or an Assembly . . . . .	427
<b>Example—Thin-Film Resistance</b>	<b>430</b>
Model Definition . . . . .	430
Results. . . . .	432
Modeling Using the Graphical User Interface . . . . .	434
Comparing the Thin-Film Approximation with the Full 3D Model . . . . .	437
<b>Using Identity Conditions</b>	<b>438</b>
Creating Identity Conditions . . . . .	438
<b>Example—Thin-Layer Diffusion</b>	<b>442</b>
Modeling Using the Graphical User Interface . . . . .	443

# Chapter 16: Deformed Meshes

<b>Deformed Mesh Fundamentals</b>	<b>454</b>
Mathematical Description of the Mesh Movement . . . . .	454
Derivatives . . . . .	456
<b>Frames for Deformed Meshes</b>	<b>458</b>
Frames in the Model Navigator . . . . .	458
Settings for Frames . . . . .	462
Frame Selection for Coupling Variables . . . . .	462
<b>The Moving Mesh Application Mode</b>	<b>463</b>
Subdomain Settings . . . . .	463
Boundary Conditions . . . . .	465
Smoothing Methods. . . . .	466
Remeshing . . . . .	466
<b>Example—Electrochemical Polishing</b>	<b>476</b>
Introduction . . . . .	476
Model Definition . . . . .	476
Modeling in COMSOL Multiphysics . . . . .	477
Results. . . . .	477
Modeling Using the Graphical User Interface . . . . .	478
<b>The Parameterized Geometry Application Mode</b>	<b>483</b>
Subdomain Settings . . . . .	483
Boundary Conditions . . . . .	484
Point Settings . . . . .	486
<b>Example—Eigenmode Analysis of an L-Shaped Membrane with Rounded Corner</b>	<b>487</b>
Introduction . . . . .	487
Model Definition . . . . .	487
Modeling in COMSOL Multiphysics . . . . .	487
Results. . . . .	488
Reference . . . . .	488
Modeling Using the Graphical User Interface . . . . .	489

<b>Retrieving the Deformed Mesh</b>	<b>493</b>
<b>Limitations of the Deformed Mesh Feature</b>	<b>494</b>

## Chapter 17: Stabilization Techniques

<b>Numerical Stabilization</b>	<b>496</b>
Stabilization—An Example . . . . .	497
<b>Artificial Diffusion Types</b>	<b>500</b>
Isotropic Diffusion . . . . .	500
Streamline Diffusion. . . . .	501
Crosswind Diffusion . . . . .	504
References . . . . .	505
<b>INDEX</b>	<b>507</b>



# Introduction

This *COMSOL Multiphysics Modeling Guide* provides an in-depth examination of the application modes in COMSOL Multiphysics® and how to use them to model different types of physics and to perform equation-based modeling using PDEs.

This chapter gives an overview of the available application modes as well as some general guidelines for effective modeling.

# Overview of the COMSOL Multiphysics Application Modes

Solving PDEs generally means you must take the time to set up the underlying equations, material properties, and boundary conditions for a given problem. COMSOL Multiphysics, however, relieves you of much of this work. The package provides a number of *application modes* that consist of predefined templates and user interfaces already set up with equations and variables for specific areas of physics. Special properties allow the selection of, for instance, analysis type and model formulations. The application mode interfaces consist of customized dialog boxes for the physics in subdomains and on boundaries, edges, and points along with predefined PDEs. A set of application-dependent variables makes it easy to visualize and postprocess the important physical quantities using conventional terminology and notation. Adding even more flexibility, the *equation system view* provides the possibility to examine and modify the underlying PDEs in the case where a predefined application mode does not exactly match the application you want to model.

---

**Note:** Suites of application modes that are optimized for specific disciplines together with large model libraries are available in a group of optional products: the *AC/DC Module*, *Acoustics Module*, *Chemical Engineering Module*, *Earth Science Module*, *Heat Transfer Module*, *MEMS Module*, *RF Module*, and *Structural Mechanics Module*.

---

## *Application Modes in COMSOL Multiphysics*

---

The following table lists the application modes in COMSOL Multiphysics and their availability for 1D, 1D axisymmetric, 2D, 2D axisymmetric, and 3D geometries:

APPLICATION MODE	1D	1D AXI	2D	2D AXI	3D
<b>Acoustics</b>					
Acoustics			√	√	√
<b>Diffusion</b>					
Convection and Diffusion	√	√	√	√	√

APPLICATION MODE	ID	ID AXI	2D	2D AXI	3D
Diffusion	√	√	√	√	√
<b>Electromagnetics</b>					
AC Power Electromagnetics			√		
Conductive Media DC			√	√	√
Electrostatics			√	√	√
Magnetostatics			√	√	√
<b>Heat Transfer</b>					
Convection and Conduction	√	√	√	√	√
Conduction	√	√	√	√	√
<b>Fluid Dynamics</b>					
Incompressible Navier-Stokes			√	√	√
<b>Structural Mechanics</b>					
Plane Strain			√		
Plane Stress			√		
Axial Symmetry, Stress-Strain				√	
Solid, Stress-Strain					√
<b>Deformed Mesh</b>					
Moving Mesh (ALE)	√	√	√	√	√
Parameterized Geometry			√		
<b>Optimization and Sensitivity</b>					
Optimization*	√	√	√	√	√
Sensitivity Analysis	√	√	√	√	√
<b>PDE Modes</b>					
Coefficient Form	√		√		√
General Form	√		√		√
Weak Form, Subdomain	√		√		√
Weak Form, Boundary	√		√		√
Weak Form, Edge					√
Weak Form, Point			√		√
<b>Classical PDEs</b>					
Convection-Diffusion Equation	√		√		√
Laplace's Equation	√		√		√

APPLICATION MODE	ID	ID AXI	2D	2D AXI	3D
Heat Equation	√		√		√
Helmholtz Equation	√		√		√
Poisson's Equation	√		√		√
Schrödinger Equation	√		√		√
Wave Equation	√		√		√

\* This application mode is available only if your COMSOL installation includes a valid license for the Optimization Lab.

As the table indicates, these application modes fall into five broad categories:

### THE PHYSICS MODES

Use the physics modes to instantly access convenient templates for specific application areas. Here you can specify physical properties for models in fields such as structural mechanics, heat transfer, and electromagnetics. Details on how to use the physics modes appear in the section “Using the Physics Modes” on page 13.

### THE DEFORMED MESH APPLICATION MODES

These application modes provide support for applications with moving boundaries using the Moving Mesh (ALE) application mode and for parameterized geometries in 2D. See the chapter “Deformed Meshes” on page 453 for more information.

### THE OPTIMIZATION AND SENSITIVITY APPLICATION MODES

The Sensitivity Analysis application mode adds sensitivity analysis to any type of multiphysics model. See “Sensitivity Analysis” on page 291 for more information.

The Optimization application mode provides functionality for combining multiphysics modeling with optimization. See the chapter “Optimization” on page 313 for details.

---

**Note:** The Optimization application mode requires the Optimization Lab.

---

### THE PDE MODES

Turn to these modes to model directly with PDEs when you cannot find a suitable physics mode. With these modes you define the problem in terms of mathematical expressions and coefficients.



COMSOL Multiphysics includes three PDE modes:

- The *Coefficient form* allows you to solve linear or almost linear problems using PDEs and coefficients that often correspond directly to various physical properties.
- The *General form* provides a computational framework specialized for highly nonlinear problems. Consider using a weak form for these problems, too.
- The *Weak form* makes it possible to model a wider class of problems, for example models with mixed time and space derivatives, or models with phenomena on boundaries, edges, or points as described with PDEs. In terms of convergence rate, these modes also set a computational framework suited for all types of nonlinear problems.

For details on how to apply the PDE modes, see the sections “PDE Modes for Equation-Based Modeling” on page 245 and “The Weak Form” on page 349.

### **CLASSICAL PDES**

The Classical PDEs folder contain application modes that describe a suite of well-known PDEs. They are special cases of the coefficient form PDE and are not meant to serve as templates.

#### *Selecting an Application Mode*

---

When creating a model in COMSOL Multiphysics, you can select a single application mode that describes one type of physics or select several application modes for multiphysics modeling and coupled-field analyses.

### **MODELING USING A SINGLE APPLICATION MODE**

Most of the physics application modes contain stationary, eigenvalue, and dynamic (time-dependent) analysis types. As already mentioned, these application modes provide modeling interfaces where you can create models using material properties, boundary conditions, and initial conditions. Each of the application modes comes with a template that automatically supplies the appropriate underlying PDE.

If you cannot find a physics mode that matches a given problem, try one of the PDE modes, which allow you to define a custom model in general mathematical terms. Indeed, COMSOL Multiphysics can model virtually any scientific phenomenon or engineering problem that originates from the laws of science.

## MULTIPHYSICS MODELING USING MULTIPLE APPLICATION MODES

When modeling real-world systems, you often need to include the interaction between different kinds of physics. For instance, the properties of an electronic component such as an inductor vary with temperature. To solve such a problem, combine two or several application modes into a single model using the multiphysics features of COMSOL Multiphysics. For the example just mentioned, combine the Conductive Media DC and Heat Transfer by Conduction application modes. In this way you create a system of two PDEs with two dependent variables:  $V$  for the electric potential and  $T$  for the temperature. There are also predefined multiphysics couplings that provide two or more coupled application modes for some common multiphysics applications.

Combining physics modes and PDE modes also works well. Assume, for instance, that you want to model the fluid-structure interactions due to the vibrations of yoghurt in a cardboard container as it rides on a conveyor belt. You could start with the Plane Stress mode for structural mechanics to model the container walls and then add a PDE to model the irrotational flow of the fluid. This approach also creates a system of two PDEs but requires that you define one of them from scratch, in this case using the general PDE formulation.

To summarize the proposed strategy for modeling processes that involve several types of physics: Look for application modes suitable for the phenomena of interest. If you find them among the physics modes, use them; if not, add one or more PDE modes.

The approach when coupling multiple application modes is to use the dependent variables or their derivatives directly, or to use expressions containing the dependent variables. The coupling can occur in subdomains and on boundaries.

# Modeling Guidelines

To allow you to model large-scale problems, COMSOL Multiphysics lets you tune solver settings and use symmetries and other model properties to reach a solution or—failing that—interrupt the solution process to retrieve a partial solution.

## *Using Symmetries*

---

By using symmetries in a model you can reduce its size by one-half or more, making this an efficient tool for solving large problems. This applies to the cases where the geometries and modeling assumptions include symmetries.

The most important types of symmetries are axial symmetry and symmetry and antisymmetry planes or lines:

- *Axial Symmetry* is common for cylindrical and similar 3D geometries. If the geometry is axisymmetric, there are variations in the radial ( $r$ ) and vertical ( $z$ ) direction only and not in the angular ( $\theta$ ) direction. You can then solve a 2D problem in the  $rz$ -plane instead of the full 3D model, which can save considerable memory and computation time. Many COMSOL Multiphysics application modes are available in axisymmetric versions.
- *Symmetry and Antisymmetry Planes or Lines* are common in both 2D and 3D models. *Symmetry* means that a model is identical on either side of a dividing line or plane. For a scalar field, the normal flux is zero across the symmetry line. In structural mechanics, the symmetry conditions are different. *Antisymmetry* means that the loading of a model is oppositely balanced on either side of a dividing line or plane. For a scalar field, the dependent variable is 0 along the antisymmetry plane or line. Structural mechanics applications have other antisymmetry conditions. Many application modes have symmetry conditions directly available in the **Boundary Settings** dialog box.

To take advantage of symmetry planes and symmetry lines, all of the geometry, material properties, and boundary conditions must be symmetric, and any loads or sources must be symmetric or antisymmetric. You can then build a model of the symmetric portion, which can be half, a quarter, or an eighth of the full geometry, and apply the appropriate symmetry (or antisymmetry) boundary conditions.

Especially in 3D modeling, extensive memory usage dictates some extra precautions. First, check that you have selected an iterative linear system solver. Normally you do not need to worry about which solver to use, because the application mode makes an appropriate default choice. In some situations, though, it might be necessary to make additional changes to the solver settings and the model.

### **ESTIMATING THE MEMORY USE FOR A MODEL**

Out-of-memory messages occur when COMSOL Multiphysics tries to allocate an array that does not fit sequentially in memory. It is common that the amount of available memory seems large enough for an array, but there might not be a contiguous block of that size due to memory fragmentation. The only solver that requires a single large contiguous memory block is the UMFPACK direct solver.

In estimating how much memory it takes to solve a specific model, the following factors are the most important:

- The number of node points
- The number of dependent and independent variables
- The element order
- The sparsity pattern of the system matrices. The sparsity pattern, in turn, depends on the shape of the geometry and the mesh. For example, an extended ellipsoid gives sparser matrices than a sphere.

The PARDISO out-of-core solver can make use of available disk space to solve large models that do not fit in the available memory.

### **CREATING A MEMORY-EFFICIENT GEOMETRY**

A first step when dealing with large models is to try to reduce the model geometry as much as possible. Often you can find symmetry planes and reduce the model to a half, a quarter or even an eighth of the original size. Memory usage does not scale linearly but rather polynomially ( $Cn^k$ ,  $k > 1$ ), which means that the model needs less than half the memory if you find a symmetry plane and cut the geometry size by half. Other ways to create a more memory-efficient geometry include:

- Avoiding small geometry objects where not needed and using Bézier curves instead of polygon chains.

- Making sure that the mesh elements are of a high quality. Mesh quality is important for an iterative linear system solver. Convergence is faster and more robust if the element quality is high.
- Avoiding geometries with sharp, narrow corners. Mesh elements get thin when they approach sharp corners, leading to poor element quality in the adjacent regions.

### *Selecting an Element Type*

---

As the default element type for most application modes, COMSOL Multiphysics uses second-order Lagrange elements or shape functions (see “Finite Elements” on page 482 of the *COMSOL Multiphysics Reference Guide* for an overview of the available element types). These and other higher-order elements add additional degrees of freedom on midpoint and interior nodes in the mesh elements. These added degrees of freedom provide a more accurate solution but also require more memory due to the reduced sparsity of the discretized system. For many application areas, such as stress analysis in structural and solid mechanics, the increased accuracy of a second-order element is important. In fluid-flow modeling using the incompressible Navier-Stokes equations, a combination of element types using an element for the velocity components of a higher order than that for the pressure usually provides the best result. The default element for the Incompressible Navier-Stokes application mode is the P2-P1 element using second-order elements for the velocity components and linear elements for the pressure. For other applications you can select a first-order element instead of a second-order element, or reduce the element order in general, to reduce memory use.

### *Analyzing Model Convergence and Accuracy*

---

It is important that the finite element model accurately captures local variations in the solution such as stress concentrations. In some cases you can compare your results to values from handbooks, measurements, or other sources of data. Many examples in the *COMSOL Multiphysics Modeling Guide* and the *COMSOL Multiphysics Model Library* include comparisons to established results or analytical solutions. Look for these *benchmark models* as a means of checking results.

If a model has not been verified by other means, a convergence test is useful for determining if the mesh density is sufficient. Here you refine the mesh and run the analysis again, and then you see if the solution is converging to a stable value as the mesh is refined. If the solution changes when you refine the mesh, the solution is mesh dependent, so the model requires a finer mesh. You can use adaptive mesh refinement

(see “Avoiding Inverted Mesh Elements” on page 374 of the *COMSOL Multiphysics User’s Guide*), which adds mesh elements based on an error criterion to resolve those areas where the error is large.

For convergence, it is important to avoid singularities in the geometry (see “Avoiding Singularities and Degeneracies in the Geometry” on page 94 of the *COMSOL Multiphysics User’s Guide* for more information).

### *Achieving Convergence When Solving Nonlinear Equations*

---

Nonlinear problems are often difficult to solve. In many cases, no unique solution exists. COMSOL Multiphysics uses a Newton-type iterative method to solve nonlinear systems of PDEs. This solution method can be sensitive to the initial estimate of the solution. If the initial conditions are too far from the desired solution, convergence might be impossible, even though it might be simple from a different starting value.

You can do several things to improve the chances for finding the relevant solutions to difficult nonlinear problems:

- Provide the best possible initial values.
- Solve sequentially and iterate between single-physics equations; finish by solving the fully coupled multiphysics problem when you have obtained better starting guesses.
- Ensure that the boundary conditions are consistent with the initial solution and that neighboring boundaries have compatible conditions that do not create singularities.
- Refine the mesh in regions of steep gradients.
- For convection-type problem, introduce artificial diffusion to improve the problem’s numerical properties (see “Stabilization Techniques” on page 495). Most application modes for fluid flow provide artificial diffusion as part of the default settings.
- Scaling can be an issue when one solution component is zero. In those cases, the automatic scaling might not work. See “Scaling of Variables and Equations” on page 529 of the *COMSOL Multiphysics Reference Guide* for more information.
- Turn a stationary nonlinear PDE into a time-dependent problem. Making the problem time dependent generally results in smoother convergence. By making sure to solve the time-dependent problem for a time span long enough for the solution to reach a steady state, you solve the original stationary problem.
- Use the parametric solver and vary a material property or a PDE coefficient starting from a value that makes the equations less nonlinear to the value at which you want

to compute the solution. This way you solve a series of increasingly difficult nonlinear problems. The solution of a slightly nonlinear problem that is easy to solve serves as the initial value for a more difficult nonlinear problem.

### *Avoiding Strong Transients*

---

If you start solving a time-dependent problem with initial conditions that are inconsistent, or if you use boundary settings or subdomain settings that switch instantaneously at a certain time, you induce strong transient signals in a system. The time-stepping algorithm then takes very small steps to resolve the transient, and the solution time might be very long, or the solution process might even stop. Stationary problems can run into mesh-resolution issues such as overshooting and undershooting of the solution due to infinite flux problems.

Unless you want to know the details of these transients, start with initial conditions that lead to a consistent solution to a stationary problem. Only then turn on the boundary values, sources, or driving fluxes over a time interval that is realistic for your model.

In most cases you should turn on your sources using a smoothed step over a finite time. What you might think of as a step function is, in real-life physics, often a little bit smoothed because of inertia. The step or switch does not happen instantaneously. Electrical switches take milliseconds, and solid-state switches take microseconds. See “Specifying Discontinuous Functions” on page 149 of the *COMSOL Multiphysics User’s Guide* for information about smoothed step functions to try out.

### *Typographical Conventions*

---

All COMSOL manuals use a set of consistent typographical conventions that should make it easy for you to follow the discussion, realize what you can expect to see on the screen, and know which data you must enter into various data-entry fields. In particular, you should be aware of these conventions:

- A **boldface** font of the shown size and style indicates that the given word(s) appear exactly that way on the COMSOL graphical user interface (for toolbar buttons in the corresponding tooltip). For instance, we often refer to the **Model Navigator**, which is the window that appears when you start a new modeling session in COMSOL; the corresponding window on the screen has the title **Model Navigator**. As another example, the instructions might say to click the **Multiphysics** button, and

the boldface font indicates that you can expect to see a button with that exact label on the COMSOL user interface.

- The names of other items on the graphical user interface that do not have direct labels contain a leading uppercase letter. For instance, we often refer to the Draw toolbar; this vertical bar containing many icons appears on the left side of the user interface during geometry modeling. However, nowhere on the screen will you see the term “Draw” referring to this toolbar (if it were on the screen, we would print it in this manual as the **Draw** menu).
- The symbol > indicates a menu item or an item in a folder in the **Model Navigator**. For example, **Physics>Equation System>Subdomain Settings** is equivalent to: On the **Physics** menu, point to **Equation System** and then click **Subdomain Settings**. **COMSOL Multiphysics>Heat Transfer>Conduction** means: Open the **COMSOL Multiphysics** folder, open the **Heat Transfer** folder, and select **Conduction**.
- A Code (monospace) font indicates keyboard entries in the user interface. You might see an instruction such as “Type 1.25 in the **Current density** edit field.” The monospace font also indicates COMSOL Script codes.
- An *italic* font indicates the introduction of important terminology. Expect to find an explanation in the same paragraph or in the Glossary. The names of books in the COMSOL documentation set also appear using an italic font.



## Using the Physics Modes

This chapter provides an overview of the physics application modes within COMSOL Multiphysics. It goes on to describe how these application modes make modeling various types of physics easier, faster, and more efficient. Later chapters review each physics mode in detail, including step-by-step examples of how to build and solve models for that particular physics application.

# The Physics Modes

One convenient way to solve a physics problem is to set it up with the assistance of templates that COMSOL Multiphysics provides in the form of its *physics modes*. The software computes the PDE coefficients based on application-specific parameters and material properties that you supply, such as Young's modulus for a structural-mechanics model or the heat capacity for a heat-transfer model.

## *Defining the Physics for a Model*

---

When working with a physics mode, you have access to all necessary settings for material properties, boundary conditions, and other modeling parameters through the **Physics** menu.

### **SUBDOMAIN SETTINGS AND MATERIAL PROPERTIES**

Enter application-specific subdomain properties by choosing **Subdomain Settings** from the **Physics** menu or double-clicking a subdomain in Subdomain mode, which you enter by opening the **Subdomain Settings** dialog box, by clicking the **Subdomain Mode** button on the Main toolbar, or on the **Physics** menu by first pointing to **Selection Mode** and then clicking **Subdomain Mode**. For a complete description of the material properties available within each application mode, see the following chapters.

### **BOUNDARY SETTINGS**

To set up boundary conditions, choose **Boundary Settings** from the **Physics** menu or double-click a boundary in Boundary mode, which you enter by opening the **Boundary Settings** dialog box, by clicking the **Boundary Mode** button on the Main toolbar, or on the **Physics** menu by first pointing to **Selection Mode** and then click **Boundary Mode**. The **Boundary Settings** dialog box is slightly different for application mode and provides the appropriate boundary conditions. For a complete description of the boundary conditions available for the physics modes, see the following chapters.

### **POINT SETTINGS**

Some models include point sources or use a point in the geometry to fix the value of a variable. To access the point settings, choose **Point Settings** from the **Physics** menu or double-click a point in Point mode, which you enter by opening the **Point Settings** dialog box, by clicking the **Point Mode** button on the Main toolbar, or on the **Physics**

menu by first pointing to **Selection Mode** and then click **Point Mode**. The **Point Settings** dialog box is not available in all application modes.

### POSTPROCESSING USING APPLICATION MODE VARIABLES

With the postprocessing mode you can visualize any variable or expression of variables by working with the **Plot Parameters**, **Cross-Section Plot Parameters**, and **Domain Plot Parameters** dialog boxes. You can also have COMSOL Multiphysics compute integrated values with the help of the **Subdomain Integration** and **Boundary Integration** dialog boxes.

Each application mode provides a special set of application-specific variables called *application mode variables*. Lists of these variables for each application mode appear in the following chapters.

---

**Note:** For all application modes you have access to shape-function variables and equation variables for the corresponding PDE solution form (coefficient, general, or weak). For further details, please refer to “Using Variables and Expressions” on page 138 of the *COMSOL Multiphysics User’s Guide*.

---

## The Physics Modes

The following table lists the available physics modes:

TABLE 2-1: THE PHYSICS MODES

APPLICATION MODE	DEFAULT Suffix	DEPENDENT VARIABLES	ANALYSIS CAPABILITIES			
			STATIONARY	TIME DEPENDENT	TIME HARMONIC	EIGENFREQUENCY
<b>ACOUSTICS</b>						
Acoustics	aco	$p$			√	√
<b>HEAT TRANSFER</b>						
Conduction	ht	$T$	√	√		
Convection and Conduction	cc	$T$	√	√		
<b>DIFFUSION</b>						

TABLE 2-1: THE PHYSICS MODES

APPLICATION MODE	DEFAULT Suffix	DEPENDENT VARIABLES	ANALYSIS CAPABILITIES			
			STATIONARY	TIME DEPENDENT	TIME HARMONIC	EIGENFREQUENCY
Convection and Diffusion	cd	$c$	√	√		
Diffusion	di	$c$	√	√		
<b>ELECTROMAGNETICS</b>						
AC Power Electromagnetics	qa	$A_z$			√	
Conductive Media DC	dc	$V$	√			
Electrostatics	es	$V$	√			
Magnetostatics	qa	$A_z$	√			
<b>STRUCTURAL MECHANICS</b>						
Solid, Stress-Strain	sld	$u, v, w$	√	√		√
Plane Stress	ps	$u, v$	√	√		√
Plane Strain	pn	$u, v$	√	√		√
Axial Symmetry, Stress-Strain	axi	$u$ or $w$	√	√		√
<b>INCOMPRESSIBLE NAVIER STOKES</b>						
Incompressible Navier-Stokes, 2D	ns	$u, v, p$	√	√		
Incompressible Navier-Stokes, 3D	ns	$u, v, w, p$	√	√		

### *Physics Mode Documentation*

Chapters 3–8 in this manual give detailed descriptions of each physics mode, typically broken down into the following sections:

#### **VARIABLES AND SPACE DIMENSIONS**

The *Variables and Space Dimensions* section describes the dependent and independent variables or space dimensions (1D, 2D, 3D, or axisymmetric) you can use in the application mode.

## **PDE FORMULATION**

The *PDE Formulation* section presents the equations the physics mode solves. It also lists the material properties, sources, and coefficients in the COMSOL Multiphysics formulation.

## **APPLICATION SCALAR VARIABLES**

The *Application Scalar Variables* section lists nongeometric variables specific to the application mode. The default values are physical constants such as the permittivity of vacuum or arbitrary values, for example, the frequency 50 Hz for the AC Power Electromagnetics application mode. Not every application mode has application scalar variables.

## **SUBDOMAIN SETTINGS**

The *Subdomain Settings* section lists the subdomain properties in the application mode such as material properties and sources and sinks.

## **BOUNDARY CONDITIONS**

The *Boundary Conditions* section lists the available boundary conditions while explaining their physical interpretation and the quantities that you specify to define the boundary settings.

## **LINE AND POINT SETTINGS**

In some application modes there are line and point settings for defining line and point sources, pressure constraints, and other properties.

## **APPLICATION MODE VARIABLES**

The *Application Mode Variables* section lists all the variables available to you when formulating the equation and performing postprocessing. You can create functions of these variables for postprocessing and visualizing the analysis and when expressing physical properties. The **Predefined quantities** lists in the dialog boxes for visualization and postprocessing do not always include all the variables; if a variable does not appear in the list, simply type its name in the **Expression** edit field.

Application mode variables are defined on the following geometric domains:

- Subdomains: S
- Boundary segments: B
- Edges: E
- Points: P

In the tables that review the application mode variables for each case, the Domain column indicates the top level where the variables are defined. Many variables that are available on subdomains are also available on boundaries, edges, and points, but they then take the average value of the values in the subdomains around the boundary, edge, or point (for the subdomains in which the variable is present). In addition to the letters above, indicating the domains where the application mode variable is valid, a *V* indicates that it is a vector-valued variable.

In some cases the table also lists the solution form for which a variable is available:

- Coefficient form: *c*
- General form: *g*
- Weak form: *w*

The Name column in the table of application mode variables lists the variables available for use, whereas the Expression column lists the implementation of the application mode variables in terms of other variables. The italic *i* and *j* in the variable names can refer to any of the space coordinates. For example,  $T_i$  can mean either  $T_x$ ,  $T_y$ , or  $T_z$  in 3D where the space coordinates are  $x$ ,  $y$ , and  $z$ . In 2D axisymmetry,  $T_i$  stands for either  $T_r$  or  $T_z$ . Further, you construct the variable names of vector components and tensor components using the names of the space coordinates. For example, if an application mode uses  $x_1$ ,  $y_1$ ,  $z_1$ , then the variables for the vector components of the temperature gradient are  $T_{x1}$ ,  $T_{y1}$ ,  $T_{z1}$ .

---

**Note:** All application mode variables include a suffix indicating which application mode they belong to. Table 2-1 on page 15 lists the default suffix for the physics modes. For example, the default suffix added to application mode variable names for an Incompressible Navier-Stokes application mode is `_ns`.

---

---

**Note:** The default space coordinate names are  $x$ ,  $y$ ,  $z$  for Cartesian coordinate systems and  $r$ ,  $\phi$  ( $\phi$ ), and  $z$  for cylindrical coordinate systems. It is possible to change these names when defining a new geometry in the Model Navigator. The variable  $t$  represents the time for time-dependent models.

---

## Acoustics

This chapter describes how to use the Acoustics application mode for modeling and simulation of acoustics and vibrations. It consists of three major sections:

- An overview of acoustics modeling
- A description of the Acoustics application mode
- A step-by-step review of how to model a reactive muffler

# Fundamentals of Acoustics

## *What is Acoustics?*

---

Acoustics is the physics of *sound*. Sound is the sensation, as detected by the ear, of very small rapid changes in the air pressure above and below a static value. This static value is atmospheric pressure (about 100,000 pascals), which varies slowly. Associated with a sound pressure wave is a flow of energy. Physically, sound in air is a longitudinal wave where the wave motion is in the direction of the energy flow. The wave crests are the pressure maxima, while the troughs represent the pressure minima.

Sound results when the air is disturbed by some source. An example is a vibrating object, such as a speaker cone in a hi-fi system. It is possible to see the movement of a bass speaker cone when it generates sound at a very low frequency. As the cone moves forward, it compresses the air in front of it, causing an increase in air pressure. Then it moves back past its resting position and causes a reduction in air pressure. This process continues, radiating a wave of alternating high and low pressure at the speed of sound.

## *Five Standard Acoustics Problems*

---

Five standard problems or scenarios occur frequently when analyzing acoustics:

- The radiation problem—A vibrating structure (a speaker, for example) radiates sound into the surrounding space. A far-away boundary condition is necessary to model the unbounded domain.
- The scattering problem—An incident wave impinges on a body and creates a scattered wave. A far-away radiation boundary condition is necessary.
- The sound field in an interior space (such as a room)—The acoustic waves stay in a finite volume so no radiation condition is necessary.
- Coupled fluid-elastic structure interaction (structural acoustics)—If the radiating or scattering structure consists of an elastic material, then you must consider the interaction between the body and the surrounding fluid. In the multiphysics coupling, the acoustic analysis provides a load (the sound pressure) to the structural analysis, and the structural analysis provides accelerations to the acoustic analysis.
- The transmission problem—The incident sound wave propagates into a body, which can have different acoustic properties. Pressure and acceleration are continuous on the boundary.



Sound waves in a lossless medium are governed by the following equation for the (differential) pressure  $p$  (with SI unit  $\text{N/m}^2$ ):

$$\frac{1}{\rho_0 c_s^2} \frac{\partial^2 p}{\partial t^2} + \nabla \cdot \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right) = Q$$

Here  $\rho_0$  ( $\text{kg/m}^3$ ) refers to the density and  $c_s$  ( $\text{m/s}$ ) denotes the speed of sound. The *dipole source*  $\mathbf{q}$  ( $\text{N/m}^3$ ) and the *monopole source*  $Q$  ( $1/\text{s}^2$ ) are both optional. The combination  $\rho_0 c_s^2$  is called the *bulk modulus*, commonly denoted  $\beta$  ( $\text{N/m}^2$ ).

A special case is a time-harmonic wave, for which the pressure varies with time as

$$p(\mathbf{x}, t) = p(\mathbf{x}) e^{i\omega t}$$

where  $\omega = 2\pi f$  ( $\text{rad/s}$ ) is the angular frequency,  $f$  ( $\text{Hz}$ ) as usual denoting the frequency. Assuming the same harmonic time-dependence for the source terms, the wave equation for acoustic waves reduces to an inhomogeneous Helmholtz equation:

$$\nabla \cdot \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right) - \frac{\omega^2 p}{\rho_0 c_s^2} = Q \quad (3-1)$$

With the source terms removed, you can also treat this equation as an eigenvalue PDE to solve for eigenmodes and eigenfrequencies.

Typical boundary conditions are:

- Sound-hard boundaries (walls)
- Sound-soft boundaries
- Impedance boundary conditions
- Radiation boundary conditions

These are described in more detail below.

In lossy media, an additional term of first order in the time derivative needs to be introduced to model attenuation of the sound waves:

$$\frac{1}{\rho_0 c_s^2} \frac{\partial^2 p}{\partial t^2} - d_a \frac{\partial p}{\partial t} + \nabla \cdot \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right) = Q$$

The damping term is absent from the standard PDE formulations in the Acoustics application mode. However, in line with COMSOL Multiphysics' general modeling philosophy, the  $d_a$  coefficient is accessible from the user interface through the **Subdomain Settings - Equation System** dialog box.

Note also that even when the sound waves propagate in a lossless medium, attenuation frequently occurs by interaction with the surroundings at the boundaries of the system. The tutorial model “Example—Reactive Muffler” on page 35 provides an example of this.

# The Acoustics Application Mode

The Acoustics application mode in COMSOL Multiphysics is designed for the analysis of various types of acoustics problems, all concerning pressure waves in a fluid. An acoustics model can be part of a larger multiphysics model that describes, for example, the interactions between structures and acoustic waves.

## *Variables and Space Dimensions*

---

The Acoustics application mode solves for the acoustic pressure,  $p$ . It is available for 2D, 2D axisymmetric, and 3D geometries.

## *PDE Formulation*

---

COMSOL Multiphysics supplies two PDE formulations—or analysis types—for the Acoustics application mode from which you can choose:

- Time-harmonic analysis
- Eigenfrequency analysis

### **TIME-HARMONIC ANALYSIS**

The time-harmonic—or frequency-domain—formulation is based on the inhomogeneous Helmholtz equation given in Equation 3-1 on page 21 and repeated here for convenience:

$$\nabla \cdot \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right) - \frac{\omega^2 p}{\rho_0 c_s^2} = Q$$

With this formulation you can compute the frequency response using the parametric solver to sweep over a frequency range using a harmonic load.

### **EIGENFREQUENCY ANALYSIS**

In the eigenfrequency formulation the source terms are absent and you solve for the eigenmodes and the eigenvalues or eigenfrequencies:

$$\nabla \cdot \left( -\frac{1}{\rho_0} \nabla p \right) + \frac{\lambda^2 p}{\rho_0 c_s^2} = 0 \quad (3-2)$$

The eigenvalue  $\lambda$  introduced in this equation is related to the eigenfrequency,  $f$ , through  $\lambda = -i2\pi f$ .

You can switch between specifying the eigenfrequencies and the eigenvalues by choosing **Properties** from the **Physics** menu and changing the value of the property **Specify eigenvalues using** in the **Application Mode Properties** dialog box. There you can also change the analysis type.

Equations 3-1 and 3-2 are both defined in three space dimensions. Because they are given in coordinate-independent notation, the equations apply to 2D, 2D axisymmetric, and 3D models alike. However, upon expanding the  $\nabla$  operators, the extra symmetries possessed by 2D and 2D axisymmetric models imply that the equations can be further adapted to the case at hand. For this reason, the general discussion that follows immediately below applies in all its details only to the 3D case, while the particularities of the remaining cases are presented in two separate subsections.

### *Subdomain Settings*

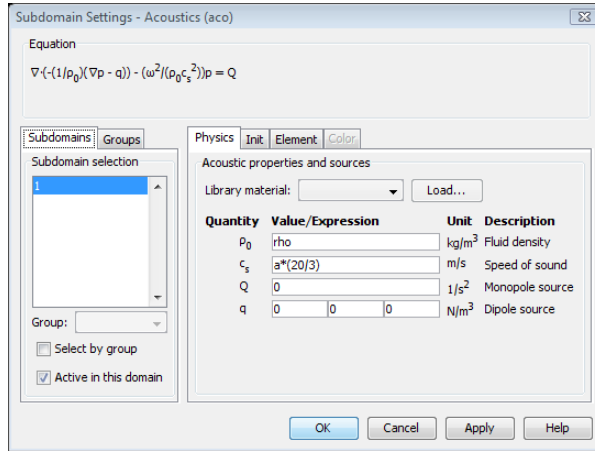
---

The application-mode characteristic quantities defined on subdomains are:

QUANTITY	VARIABLE	DESCRIPTION
$\rho_0$	rho0	Fluid density
$c_s$	cs	Speed of sound
$\mathbf{q}$	qx, qy, qz	Dipole source
$Q$	Q	Monopole source

- In the **Fluid density** edit field you enter the density of the fluid in which the acoustic waves propagate. The default value is  $1.25 \text{ kg/m}^3$ , the density of air expressed in SI units.
- The **Speed of sound** edit field takes the speed of the sound wave. The default value is  $343 \text{ m/s}$ , the speed of sound in air at approximately  $20$  degrees Celsius.
- The **Dipole source** edit fields contain the individual components of the dipole source vector,  $\mathbf{q}$ , one for each space dimension. The dipole source has the SI unit  $\text{N/m}^3$ . Its default value is  $0$ .
- The monopole source term  $Q$  in the **Monopole source** edit field has the SI unit  $1/\text{s}^2$ . It has the default value  $0$ .

You specify these properties in the **Subdomain Settings** dialog box.



## Boundary Conditions

This section describes the boundary conditions available for the Acoustics application mode and lists the analysis types to which each condition applies.

### SOUND-HARD BOUNDARIES (WALLS)

A *sound-hard boundary* is a boundary at which the normal component of the acceleration is zero:

$$-\mathbf{n} \cdot \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right) = 0$$

For zero dipole charge and constant fluid density, this means that the normal derivative of the pressure is zero at the boundary:

$$\frac{\partial p}{\partial n} = 0$$

Sound-hard boundaries are available for all analysis types.

### SOUND-SOFT BOUNDARIES

At a *sound-soft boundary* the differential pressure vanishes:

$$p = 0$$

Sound-soft boundaries are also available for all analysis types.

### PRESSURE SOURCE

This boundary condition means that you specify a constant acoustic pressure to be maintained at the boundary:

$$p = p_0$$

The pressure-source condition is available only for time-harmonic analysis.

### IMPEDANCE BOUNDARY CONDITION

The impedance boundary condition is a generalization of the sound-hard and sound-soft boundary conditions:

$$\mathbf{n} \cdot \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right) - \frac{i\omega p}{Z} = 0$$

Here  $Z$  (SI unit Pa·s/m) is the acoustic input impedance of the external domain. From a physical point of view, the acoustic input impedance is the ratio between pressure and normal particle velocity. It can be expressed in terms of the characteristic impedance inside the domain,  $Z_0 = \rho c$ , as  $Z = \zeta Z_0$ , where the dimensionless quantity  $\zeta$  is called the specific acoustic impedance.

The impedance boundary condition is a good approximation for a locally reacting surface—a surface for which the normal velocity at any point depends only on the pressure at that exact point.

Note that in the two opposite limits  $Z \rightarrow \infty$  and  $Z \rightarrow 0$ , the sound-hard and sound-soft boundary conditions are recovered. Impedance boundary conditions are available for time-harmonic and transient analysis.

### RADIATION BOUNDARY CONDITIONS

The radiation boundary conditions allow an outgoing wave to leave the modeling domain with minimal reflections. In specifying a boundary condition of this kind you have the choice between three wave types: plane, cylindrical, or spherical. You can thus adapt the condition to the geometry of your modeling domain.

The radiation boundary conditions read

$$-\mathbf{n} \cdot \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right) + (ik + \kappa(r)) \frac{p}{\rho_0} = (ik + \kappa(r) - i(\mathbf{k} \cdot \mathbf{n})) \frac{p_0}{\rho_0} e^{-i(\mathbf{k} \cdot \mathbf{r})}$$

where  $k$  is the wave number (a predefined application-mode variable; see Table 3-1 on page 30) and  $\kappa(r)$  is a function whose form depends on the wave type:

- Plane wave:  $\kappa(r) = 0$
- Cylindrical wave:  $\kappa(r) = 1/(2r)$
- Spherical wave:  $\kappa(r) = 1/r$

In the latter two cases,  $r$  is the shortest distance from the point  $\mathbf{r} = (x, y, z)$  on the boundary to the source. The right-hand side of the equation represents an optional incoming plane pressure wave with amplitude  $p_0$  and wave vector  $\mathbf{k} = k \mathbf{n}_k$ , where  $\mathbf{n}_k$  denotes the unit vector in the direction of propagation.

To define a radiation boundary condition, select a plane, cylindrical, or spherical wave from the **Wave type** list. In addition, you must specify:

- $p_0$ —the pressure source amplitude
- $\mathbf{n}_k$ —the wave direction vector
- $\mathbf{r}_0 = (x_0, y_0, z_0)$ —a point on the source axis (for a cylindrical wave) or the source location (for a spherical wave)
- $\mathbf{r}_{\text{axis}}$ —the source axis direction (only for cylindrical waves)

The default value of  $\mathbf{n}_k$  is the inward normal vector,  $-\mathbf{n}$ , which is the natural direction for waveguides and similar structures. For wave propagation in open space,  $\mathbf{k}$  can point in any direction.

---

**Note:** You do not have to normalize the vector whose components you enter in the **Wave direction** edit fields; the software explicitly normalizes the components to make  $\mathbf{n}_k$  a unit vector in the direction that you specify.

---

Radiation boundary conditions are available for time-harmonic analysis only.

#### SPECIFIED NORMAL ACCELERATION

$$-\mathbf{n} \cdot \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right) = a_n$$

The inward normal acceleration  $a_n$  represents an external source term. You can also use it for coupling your acoustics model to a structural analysis. The specified normal acceleration is available for time-harmonic analysis.

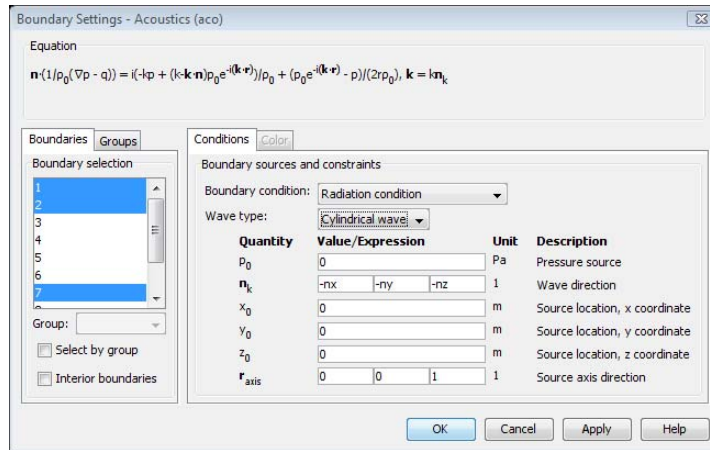
## INTERFACE CONDITIONS ON BORDERS

The set of interface conditions for interior boundaries in the Acoustics application mode is:

- *Continuity*

$$\mathbf{n} \cdot \left[ \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right)_1 - \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right)_2 \right] = 0$$

- *Sound soft boundaries*— $p = 0$
- *Pressure condition*— $p = p_0$  (not available for eigenvalue analysis)



## Scalar Variables

The following scalar variables appear in the Acoustics application mode, independently of the number of space dimensions:

QUANTITY	VARIABLE	DESCRIPTION	ANALYSIS
$f$	freq	Frequency	H
$i\omega$	iomega	Imaginary angular frequency	E
$p_{\text{ref}}$	p_ref	Reference pressure	Both

Here H and E stands for time-harmonic analysis and eigenfrequency analysis, respectively.

- *Frequency*—This is an important property for time-harmonic acoustics analysis. It is related to the angular frequency by the equation



$$\omega = 2\pi f$$

The wavelength is given by  $\lambda = c_s/f$ . Therefore, the higher the frequency, the shorter the wavelength. To resolve a wave, it is important that the mesh size be smaller than the wavelength. As a rule of thumb, use a minimum mesh resolution of a few elements per wavelength with the default second-order Lagrange elements.

Another important acoustics property is the *wave number*,  $k$ , which is defined as

$$k = \frac{2\pi}{\lambda} = \frac{\omega}{c_s}$$

- *Imaginary angular frequency*—For the eigenfrequency analysis type the quantity  $i\omega$  is used as a scalar variable in place of the frequency,  $f$ . By default, it is related to the eigenvalue,  $\lambda$ , by the equation

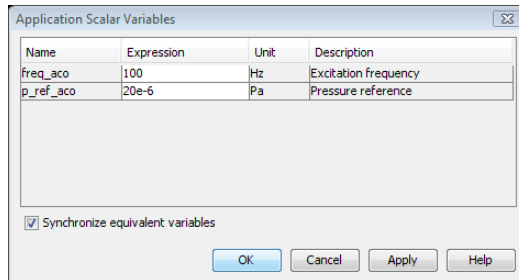
$$i\omega = -\lambda$$

(Note that the symbol  $\lambda$  is used here in a different meaning than in the previous paragraph.)

- *Reference pressure*—The zero level on the dB scale varies with the type of fluid. The default value (which is the standard value for air) is 0.02 mPa ( $20 \cdot 10^{-6}$  Pa), which then corresponds to 0 dB. This variable occurs only in calculations of the sound pressure level based on the root mean square (rms) pressure,

$$\bar{p} = \sqrt{\frac{1}{2} p \text{conj}(p)}$$

an expression valid for the case of harmonically time-varying acoustic pressure,  $p$ .



### *Application Mode Variables*

The variables in the following table are available for use in expressions and for postprocessing purposes. Almost all application-mode parameters are available as

variables. Some variables are available only in certain analysis types as indicated in the Analysis column, where an H means that the variable is available only for time-harmonic analysis. The table uses an index convention where a single index  $i$  denotes any one of the global space variables ( $x, y, z$ ) while a summation over  $i$  runs over all three space dimensions.

TABLE 3-1: ACOUSTICS APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
p	$p$	Both	All	Pressure	$p$
Lp	$L_p$	Both	S	Sound pressure level	$10 \log \left( \frac{p^2}{p_0^2} \right)^{-2}$
rho0	$\rho_0$	Both	S	Fluid density	$\rho_0$
cs	$c_s$	Both	S	Speed of sound	$c_s$
k	$k$	Both	S	Wave number	$\omega/c_s$
$q_i$	$q_i$	Both	S	Dipole source, $x_i$ component	$q_i$
normq	$ \mathbf{q} $	Both	S	Dipole source, norm	$\sqrt{\sum_i q_i^2}$
$a_i$	$a_i$	Both	S	Local acceleration, $x_i$ component	$-\frac{1}{\rho_0} \left( \frac{\partial p}{\partial x_i} - q_i \right)$
norma	$ \mathbf{a} $	Both	S	Local acceleration, norm	$\sqrt{\sum_i a_i^2}$
na	$a_n$	Both	B	Outward normal acceleration	$\mathbf{n} \cdot \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right)$
$v_i$	$v_i$	H	S	Local velocity, $x_i$ component	$a_i/(i\omega)$
normv	$ \mathbf{v} $	Both	S	Local velocity, norm	$\sqrt{\sum_i v_i^2}$
nv	$v_n$	H	B	Outward normal velocity	$a_n/(i\omega)$

TABLE 3-1: ACOUSTICS APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
$I_i$	$I_i$	Both	S	Intensity, $x_i$ component	$\text{conj}(v_i) p$
normI	$I$	Both	S	Intensity, norm	$\sqrt{\sum_i I_i^2}$

To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: `aco`), for example, `Lp_aco`. (This does not apply to the dependent variable for the pressure.)

### Units

The following table collects the SI units for the most important physical quantities in the Acoustics application mode:

QUANTITY	SYMBOL	SI UNIT	ABBREVIATION
Pressure	$p$	pascal	Pa
Density	$\rho$	kilogram/meter <sup>3</sup>	kg/m <sup>3</sup>
Frequency	$f$	hertz	Hz
Wave number	$k$	1/meter	1/m
Dipole source	$\mathbf{q}$	newton/meter <sup>3</sup>	N/m <sup>3</sup>
Monopole source	$Q$	1/second <sup>2</sup>	1/s <sup>2</sup>
Speed of sound	$c_s$	meter/second	m/s
Acoustic impedance	$Z$	pascal-second/meter	Pa·s/m
Normal acceleration	$a_n$	meter/second <sup>2</sup>	m/s <sup>2</sup>
Source location	$r_0$	meter	m
Wave direction	$\mathbf{n}_k$	(dimensionless)	1

### 2D

#### PDE FORMULATION

In 2D, the independent variables are the Cartesian coordinates  $x$  and  $y$ , while the model is assumed to be uniform in the perpendicular  $z$  direction. Under this assumption, any wave motion in the latter direction is accounted for by specifying the *out-of-plane wave number*,  $k_z$ , defined by the equation

$$(p(x, y, z, t) = p(x, y)e^{i(\omega t - k_z z)}) \quad (3-3)$$

#### Time-Harmonic Analysis

Using Equation 3-3 and expanding the 3D  $\nabla$  operators in Equation 3-1, the equation for the pressure,  $p(x, y)$ , becomes

$$\nabla \cdot \left[ -\frac{1}{\rho_0}(\nabla p - \mathbf{q}) \right] - \left[ \left( \frac{\omega}{c_s} \right)^2 - k_z^2 \right] \frac{p}{\rho_0} = Q$$

where the  $\nabla$ 's denote 2D differential operators.

#### Eigenfrequency Analysis

Identical reasoning as for the time-harmonic case leads to the equation

$$\nabla \cdot \left( -\frac{1}{\rho_0} \nabla p \right) + \left[ \left( \frac{\lambda}{c_s} \right)^2 + k_z^2 \right] \frac{p}{\rho_0} = 0$$

### SCALAR VARIABLES

In addition to the scalar variables available in the three-dimensional case, in 2D you can also specify the value of the out-of-plane wave number,  $k_z$ :

QUANTITY	VARIABLE	DESCRIPTION	ANALYSIS TYPE
$f$	freq	Frequency (time-harmonic)	H
$i\omega$	iomega	Imaginary angular frequency	E
$k_z$	kz	Out-of-plane wave number	Both
$p_{\text{ref}}$	p_ref	Reference pressure	Both

The default value of  $k_z$  is 0.

### APPLICATION MODE VARIABLES

With the interpretation that the index  $i$  stands for either  $x$  or  $y$  and that summations run over these two values, all variables listed in Table 3-1 on page 30 are defined also in the 2D case.

## 2D Axisymmetric

---

### PDE FORMULATION

In the 2D axisymmetric case, the independent variables are the radial coordinate,  $r$ , and the axial coordinate,  $z$ . The only dependence of the azimuthal coordinate,  $\varphi$ , allowed is through a phase factor:

$$p(r, \varphi, z, t) = p(r, z)e^{i(\omega t - m\varphi)} \quad (3-4)$$

where  $m$  denotes the *circumferential wave number*. Because  $\varphi$  is a periodic coordinate,  $m$  must be an integer.

#### Time-Harmonic Analysis

Using Equation 3-4 and expanding the  $\nabla$  operators in Equation 3-1, leads to the following equation for the acoustic pressure,  $p(r, z)$ :

$$\nabla \cdot \left[ -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right] - \left[ \left( \frac{\omega}{c_s} \right)^2 - \left( \frac{m}{r} \right)^2 \right] \frac{p}{\rho_0} = Q$$

#### Eigenfrequency Analysis

The corresponding eigenvalue equation reads

$$\nabla \cdot \left( -\frac{1}{\rho_0} \nabla p \right) + \left[ \left( \frac{\lambda}{c_s} \right)^2 + \left( \frac{m}{r} \right)^2 \right] \frac{p}{\rho_0} = 0$$

### BOUNDARY CONDITIONS

#### Axial Symmetry

This condition is available only for axisymmetric models, where it applies to the symmetry boundary  $r = 0$ .

### SCALAR VARIABLES

Beside the scalar variables common with the 3D and 2D cases, you can specify the value of the circumferential wave number,  $m$ , in the 2D axisymmetric case:

QUANTITY	VARIABLE	DESCRIPTION	ANALYSIS TYPE
$f$	freq	Frequency	H
$i\omega$	iomega	Imaginary angular frequency	E
$m$	m	Circumferential wave number	Both
$p_{\text{ref}}$	p_ref	Reference pressure for dB computation	Both

The default value of  $m$  is 0.

**APPLICATION MODE VARIABLES**

With the interpretation that the index  $i$  stands for either  $r$  or  $z$  and that summations run over these two values, Table 3-1 on page 30 applies also to the 2D axisymmetric case.

# Example—Reactive Muffler

## *Introduction*

---

This model examines the sound-transmission properties of an idealized reactive muffler with infinitely long inlet and outlet pipes (or a reflection-free source at the inlet pipe and a reflection-free end of the outlet pipe) and one expansion chamber. One measure of the transmission properties is the transmission-loss coefficient,  $D_{tl}$ , which is defined as

$$D_{tl} = 10 \cdot \log\left(\frac{W_i}{W_t}\right)$$

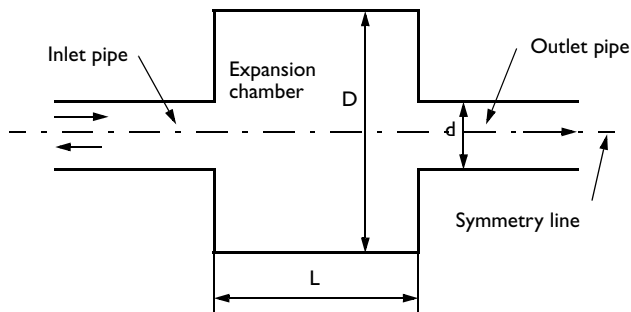
where  $W_i$  is the time-averaged incident sound power and  $W_t$  is the transmitted sound power. This problem has a theoretical 1D solution that you can compare with the FEM solution.

## *Model Definition*

---

In the following figure, a plane sound wave enters the inlet pipe (left) and is reflected and attenuated in the expansion chamber. The attenuated sound wave exits through the outlet pipe (right).

The diameter of both the inlet pipe and the outlet pipe is  $d$ , and the corresponding cross-sectional area is  $S_1$ . The expansion chamber has a diameter  $D$  with a corresponding cross-sectional area  $S_2$ .



According to Ref. 1, the 1D theoretical solution for the transmission loss to this problem is

$$D_{tl} = 10 \cdot \log \left[ 1 + \left( \frac{S_1}{2 \cdot S_2} - \frac{S_2}{2 \cdot S_1} \right)^2 \cdot (\sin(kL))^2 \right]$$

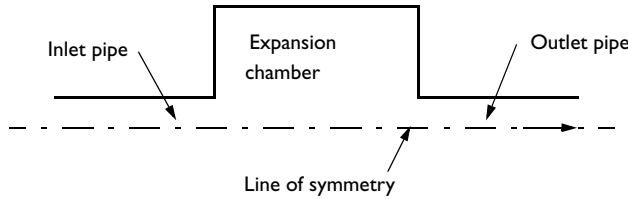
where  $k$  is the wave number;  $S_1$  and  $S_2$  are the areas of the pipes and expansion chamber; and  $L$  gives the length of the expansion chamber.

The model computes the pressure,  $p$ , for the fluid in the region defined by the above geometry. This is a time-harmonic problem so you can use the Helmholtz equation defined in the axisymmetric Acoustics application mode:

$$\nabla \cdot \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right) - \frac{\omega^2 p}{\rho_0 c_s^2} = 0$$

where  $\omega = 2\pi f$  is the angular frequency,  $\rho_0$  is the fluid density, and  $c_s$  is the speed of sound. The  $\mathbf{q}$  term is a dipole source with the dimension of force per volume.

Because this is an axisymmetric model, you need to include only half of the geometry as indicated in the following figure:



You must apply axial symmetry boundary conditions on the line of symmetry.

Assume the walls are rigid, and thus use sound-hard (wall) boundary conditions,

$$\frac{\partial p}{\partial n} = 0$$

which means that the normal derivative of the pressure is zero at the boundaries.

Radiation boundary conditions describe the inlet and outlet boundaries:

$$-\mathbf{n} \cdot \left( -\frac{1}{\rho_0} (\nabla p - \mathbf{q}) \right) + \left( \frac{ik}{\rho_0} \right) p = \frac{(ik - i(\mathbf{k} \cdot \mathbf{n})) p_0 e^{-i(\mathbf{k} \cdot \mathbf{r})}}{\rho_0}$$



The radiation boundary condition is useful when the surroundings are merely a continuation of the domain, which is the case in this model. The term on the right-hand side represents an incoming pressure wave with an amplitude  $p_0$  and a direction given by the wave vector,  $\mathbf{k}$ . In this model, an incoming pressure wave with the amplitude  $p_0 = 1$  Pa enters at the inlet boundary.

To determine the transmission loss in the model, you must first calculate the incident and transmitted time-averaged sound intensities and the corresponding sound power values. The equation

$$I = \frac{p^2}{2\rho_0 c}$$

gives the time-averaged sound intensities where  $p$  is equal to  $p_0$  at the inlet and the computed solution at the outlet.

Using the boundary integration tool, you can evaluate the incident and transmitted sound powers,  $W$ , as:

$$W = \int (I \cdot 2\pi r) dr$$

### *Results and Discussion*

---

Figure 3-1 shows the theoretical transmission loss (square markers) and the COMSOL Multiphysics solution (triangle markers) as a function of frequency. The theoretical solution has an upper frequency limit for its validity. This limit is the cut-on frequency, which defines the frequency range where only plane waves can propagate; above this frequency, also higher modes can propagate.

According to Ref. 1, the first cut-on frequency for a pipe is

$$f_{01} = 1.841 \frac{c}{\pi D}.$$

Its value in this case is approximately 332 Hz, but it is evident from the above figure that a discrepancy exists between the theoretical and the FEM solution, even below the cut-on frequency. The discrepancy also increases with frequency between the 1D theoretical model and a 3D analysis, as you can see in Ref. 1. In the lower frequency range, however, there is good agreement between the theoretical solution and the FEM solution.

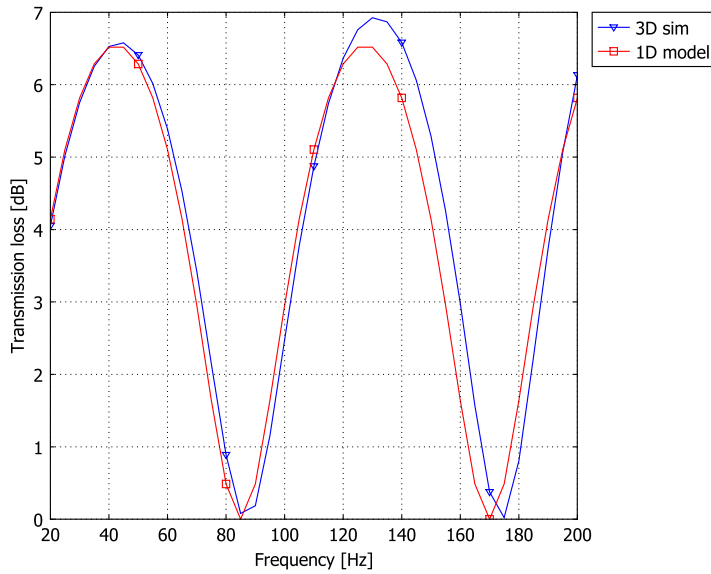


Figure 3-1: Muffler transmission loss versus frequency: theoretical solution (squares) and COMSOL Multiphysics solution (triangles).

### Reference

1. H.P. Wallin, *Ljud och Vibrationer*, Institutionen för Farkostteknik, KTH, Stockholm, Sweden, 1999 (in Swedish).

---

**Model Library path:** COMSOL\_Multiphysics/Acoustics/reactive\_muffler

---

### Modeling Using the Graphical User Interface

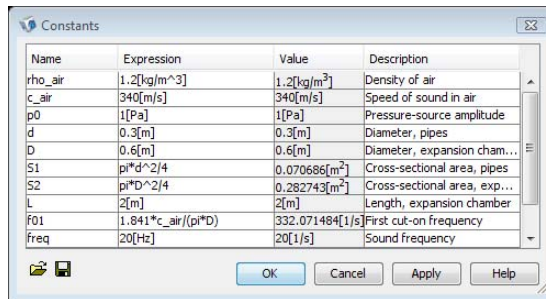
#### MODEL NAVIGATOR

- 1 Go to the **Model Navigator** and select **Axial symmetry (2D)** in the **Space dimension** list.
- 2 In the list of application modes open the **COMSOL Multiphysics>Acoustics>Acoustics** folder and then select **Time-harmonic analysis**.
- 3 Click **OK**.

## OPTIONS AND SETTINGS

- 1 Go to the **Options** menu and choose **Constants** to parameterize the model.
- 2 In the **Constants** dialog box enter the following constants, representing fluid properties and some geometrical properties to calculate the cut-on frequency and the theoretical transmission loss:

NAME	EXPRESSION	DESCRIPTION
rho_air	1.2[kg/m^3]	Density of air
c_air	340[m/s]	Speed of sound in air
p0	1[Pa]	Pressure-source amplitude
d	0.3[m]	Diameter, pipes
D	0.6[m]	Diameter, expansion chamber
S1	pi*d^2/4	Cross-sectional area, pipes
S2	pi*D^2/4	Cross-sectional area, expansion chamber
L	2[m]	Length, expansion chamber
f01	1.841*c_air/(pi*D)	First cut-on frequency
freq	20[Hz]	Sound frequency



## GEOMETRY MODELING

- 1 Shift-click the **Rectangle/Square** button to specify a rectangle.
- 2 Go to the **Rectangle** dialog box and type 0.3 in the **Width** edit field and 1 in the **Height** edit field.
- 3 Click **OK**.
- 4 Click the **Zoom Extents** button.
- 5 Shift-click the **Rectangle/Square** button to specify another rectangle.

- 6 In the **Rectangle** dialog box, type 0.6 in the **Width** edit field, 2 in the **Height** edit field, and 1 in the **z** edit field. Click **OK**.
- 7 Shift-click the **Rectangle/Square** button to specify a third rectangle.
- 8 In the **Rectangle** dialog box type 0.3 in the **Width** edit field, 1 in the **Height** edit field, and 3 in the **z** edit field. Click **OK**.
- 9 Click the **Zoom Extents** button on the Main toolbar.

## PHYSICS SETTINGS

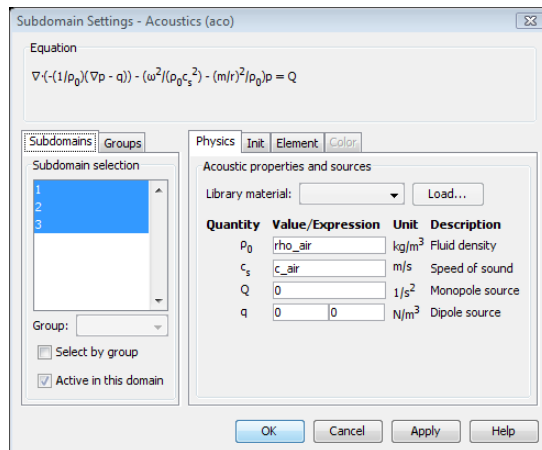
### Subdomain Settings

This model uses the fluid properties of air, specified in SI units.

Enter these quantities:

QUANTITY	ALL SUBDOMAINS
$\rho_0$	rho_air
$c_s$	c_air

- 1 From the **Physics** menu choose **Subdomain Settings**.
- 2 In the **Subdomain Settings** dialog box select all subdomains from the **Subdomain selection** list.
- 3 Type rho\_air in the **Fluid density** edit field.
- 4 Type c\_air in the **Speed of sound** edit field.
- 5 Leave the default settings (all 0) for the **Dipole source** and the **Monopole source**.
- 6 Click **OK**.

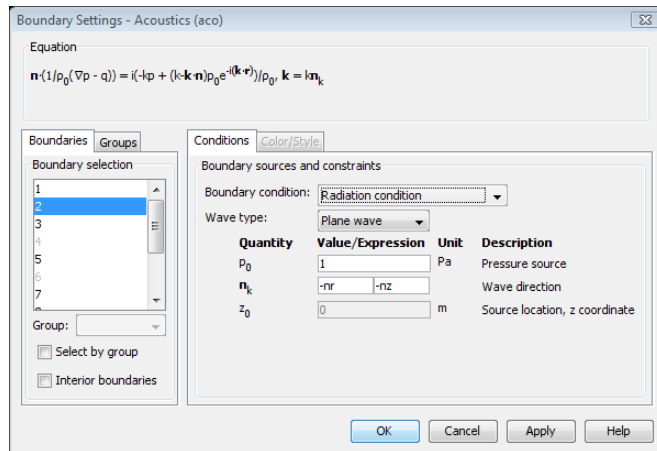


### Boundary Conditions

- 1 From the **Physics** menu choose **Boundary Settings**.
- 2 In the **Boundary Settings** dialog box enter the following boundary condition types and properties:

SETTINGS	BOUNDARIES 1, 3, 5	BOUNDARIES 8–12	BOUNDARY 2	BOUNDARY 7
Boundary condition	Axial symmetry	Sound hard boundary (wall)	Radiation condition	Radiation condition
Wave type			Plane wave	Plane wave
$P_0$			1	0

- 3 Select Boundaries 1, 3, and 5 in the **Boundary selection** list.
- 4 Select **Axial symmetry** in the **Boundary condition** list.
- 5 Select Boundaries 8–12 in the **Boundary selection** list.
- 6 Select **Sound hard boundary (wall)** in the **Boundary condition** list.
- 7 Select Boundary 2 in the **Boundary selection** list.
- 8 Select **Radiation condition** in the **Boundary condition** list.
- 9 Type 1 in the **Pressure source** edit field.

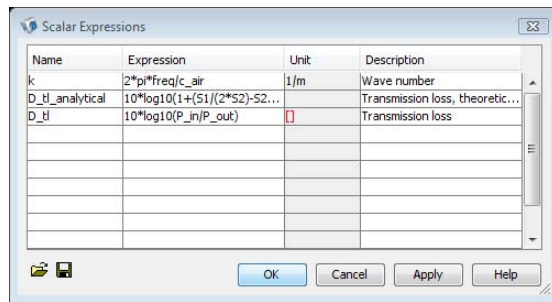


- 10 Finally select Boundary 7 in the **Boundary selection** list.
- 11 Select **Radiation condition** in the **Boundary condition** list.
- 12 Click **OK**.

### Expression Variables

- 1 On the **Options** menu, point to **Expressions**, and then click **Scalar Expressions**.
- 2 In the **Scalar Expressions** dialog box enter the following:

NAME	EXPRESSION	DESCRIPTION
k	$2\pi \cdot \text{freq} / c_{\text{air}}$	Wave number
D_t1_analytical	$10 \cdot \log_{10}(1 + (S1 / (2 \cdot S2) - S2 / (2 \cdot S1))^2 \cdot (\sin(k \cdot L))^2)$	Transmission loss, theoretical 1D model
D_t1	$10 \cdot \log_{10}(P_{\text{in}} / P_{\text{out}})$	Transmission loss



The red brackets in the **Unit** column for **D\_t1** appear because **P\_in** and **P\_out**, which you define as integration coupling variables shortly, do not have units. Because the expression is, nevertheless, dimensionally correct, you can ignore this warning.

- 3 Click **OK**.
- 4 Go to the **Options** menu and choose **Expressions** and then **Boundary Expressions**.
- 5 In the **Boundary Expressions** dialog box select **Boundary 2** from the **Boundary selection** list and enter the following boundary expression variable:

NAME	EXPRESSION
I_in	$\text{real}(\text{conj}(p_0) \cdot p_0) / (2 \cdot \rho_{\text{air}} \cdot c_{\text{air}})$

- 6 In the **Boundary Expressions** dialog box select **Boundary 7** from the **Boundary selection** list and enter the following boundary expression variable:

NAME	EXPRESSION
I_in	
I_n	$\text{real}(\text{conj}(p) \cdot p) / (2 \cdot \rho_{\text{air}} \cdot c_{\text{air}})$

7 Click **OK**.

### *Integration Coupling Variables*

1 Go to the **Options** menu and choose **Integration Coupling Variables** and then **Boundary Variables**.

2 In the **Boundary Integration Variables** dialog box select Boundary 2 and then enter the following boundary integration expression:

NAME	EXPRESSION	INTEGRATION ORDER
P_in	$I_{in} * 2 * \pi * r$	4

3 In the **Boundary Integration Variables** dialog box select Boundary 7 and enter the following boundary integration expression; when done, click **OK**.

NAME	EXPRESSION	INTEGRATION ORDER
P_in		
P_out	$I_n * 2 * \pi * r$	4

### **MESH GENERATION**

1 Go to the **Mesh** menu and choose **Free Mesh Parameters**.

2 In the **Free Mesh Parameters** dialog box, select **Finer** in the **Predefined mesh sizes** list.

3 Click **Remesh**, then click **OK**.

### **COMPUTING THE SOLUTION**

1 From the **Solve** menu, choose **Solver Parameters**.

2 In the **Solver Parameters** dialog box, select **Parametric** from the **Solver** list.

3 Type `freq` in the **Parameter name** edit field.

4 Type `20:5:200` in the **Parameter values** edit field.

5 Click **OK**.

6 Go to the **Physics** menu and choose **Scalar Variables**.

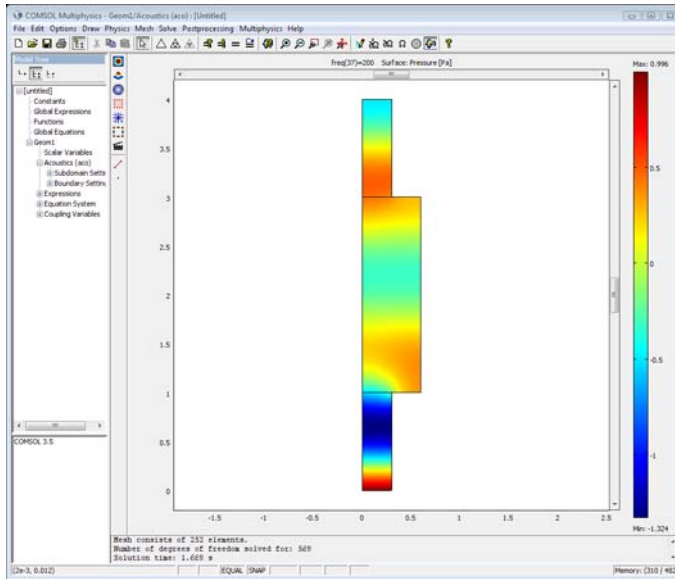
7 In the **Expression** column, type `freq` in the edit field for the frequency.

8 Click **OK**.

9 Click the **Solve** button to start the simulation.

### **POSTPROCESSING AND VISUALIZATION**

The default visualization plots the magnitude of the pressure field at the final frequency (200 Hz).



Next generate the transmission-loss plots in Figure 3-1.

- 1 Go to the **Postprocessing** menu and select **Domain Plot Parameters**.
- 2 On the **General** page, select all frequencies from the **Solutions to use** list in the **Domain Plot Parameters** dialog box.
- 3 Select the **Keep current plot** check box.
- 4 On the **Point** page, select Point 1 from the **Point selection** list.
- 5 Type  $D\_t1$  in the **Expression** edit field.
- 6 Click the **Line Settings** button and select **Triangle** in the **Line marker** list. Click **OK**.
- 7 Click **Apply** in the **Domain Plot Parameters** dialog box.  
 To make it easy to compare the two solutions, plot the theoretical solution in the same figure.
- 8 Type  $D\_t1\_analytical$  in the **Expression** edit field.
- 9 Click the **Line Settings** button. Select **Color** from the **Line color** list and **Square** from the **Line marker** list. Click **OK**.
- 10 Click **OK** in the **Domain Plot Parameters** dialog box
- 11 In the figure window, click the **Edit Plot** toolbar button. Finish the plot by editing the plot title and axis labels, and adding labels.



# 4

## Diffusion

This chapter explains how to use both the Diffusion application mode and the Diffusion and Convection application mode to model and simulate various types of transport problems. It concludes with a step-by-step example that introduces the concept of effective diffusivity in porous media.

# The Diffusion Application Mode

The Diffusion application mode treats the mass transport mechanism with the same name, which is the simplest one to describe in mathematical terms. This application mode is available in 1D, 2D, and 3D as well as in axisymmetric formulations in 1D and 2D.

The dependent variable is the mass concentration,  $c$ .

---

**Note:** The optional Chemical Engineering Module provides an application mode for the diffusion of several species as well as an application mode for problems that involve potential flow. It also supports modeling using pseudo-2D and pseudo-3D geometries.

---

## *PDE Formulation*

---

Heat transfer is a diffusion process, so the generic diffusion equation has the same structure as the heat equation. Diffusion is governed by the equation

$$\delta_{ts} \frac{\partial c}{\partial t} + \nabla \cdot (-D \nabla c) = R$$

where  $c$  is the concentration,  $D$  is the *diffusion coefficient*, and  $R$  is a reaction rate. The diffusion process can be anisotropic, in which case  $D$  is a tensor.

## *Subdomain Settings*

---

The diffusion equation coefficients to specify on the subdomains are:

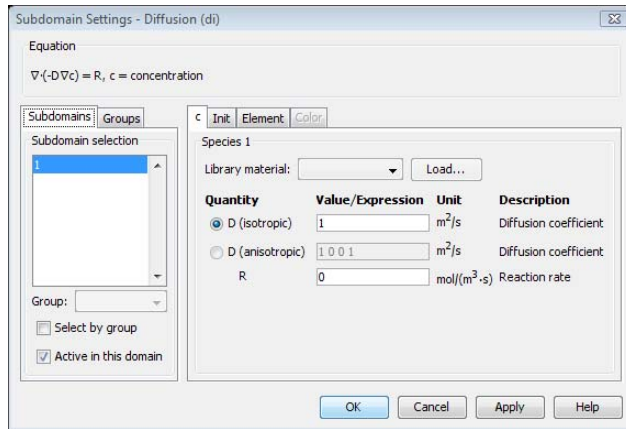
COEFFICIENT	VARIABLE	DESCRIPTION
$\delta_{ts}$	Dts_c	Time-scaling coefficient
$D$	D_c	Diffusion coefficient, diffusion coefficient tensor
$D_{ij}$	$Dx_i x_j$ _c	Diffusion coefficient tensor, $x_i x_j$ component
$R$	R_c	Reaction rate

**Time-scaling Coefficient** This coefficient is normally 1. If desired, you can change the time scale, for example, from seconds to minutes by setting it to 1/60.

**Diffusion Coefficient** This material property, denoted by  $D$ , describes a material's diffusive conductivity, in other words the relation between the concentration gradient and the mass flux.

**Reaction Rate** The reaction rate describes the volume density of mass creation. The reaction rate can be nonlinear.

You specify the equation coefficients in the **Subdomain Settings** dialog box.



For equations in 2D or 3D, pay special attention to the isotropic diffusion coefficient,  $D$ . If you select this coefficient, the application mode expands it to the diagonal of the diffusion coefficient tensor, that is,  $Dx_i x_i$  equals  $D$ .

### Boundary Conditions

The boundary conditions are:

BOUNDARY CONDITION	DESCRIPTION
$c = c_0$	Concentration
$-\mathbf{n} \cdot (-D \nabla c) = N_0 + k_c (c_b - c)$	Flux
$\mathbf{n} \cdot (-D \nabla c) = 0$	Insulation/Symmetry

BOUNDARY CONDITION	DESCRIPTION
$\mathbf{n}_1 \cdot (-D\nabla c)_1 = \frac{D}{d}(c_1 - c_2)$	Thin boundary layer
$\mathbf{n}_2 \cdot (-D\nabla c)_2 = \frac{D}{d}(c_2 - c_1)$	
$\mathbf{n} \cdot (\mathbf{N}_1 - \mathbf{N}_2) = 0$	Continuity
$-\mathbf{n} \cdot (\mathbf{N}_1 - \mathbf{N}_2) = N_0$	Flux discontinuity
$\mathbf{n} \cdot (-D\nabla c) = 0$	Axial symmetry

**Concentration** In the equation for the concentration boundary condition,  $c_0$  is a user-specified concentration.

**Flux** In the equation for the flux condition,  $N_0$  is an arbitrary user-specified flux expression. Furthermore,  $k_c$  represents the mass transfer coefficient and  $c_b$  is the bulk concentration in a fictitious diffusion layer at the boundary.

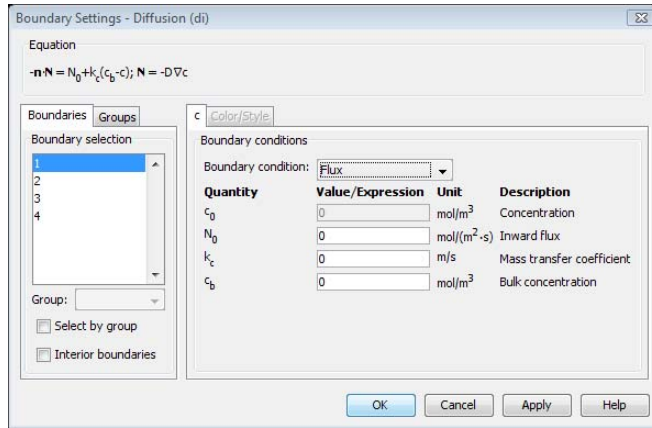
**Thin boundary layer** You can use the thin boundary layer condition to model a thin layer of a material with a small diffusion coefficient compared to the adjacent domains. The layer has the thickness  $d$  and the diffusion coefficient  $D$ . This boundary condition is only available at the border between the parts in an assembly.

**Continuity** This is the default boundary condition on interior boundaries and pair boundaries; it is not applicable to exterior boundaries.

**Flux discontinuity** This boundary condition represents a discontinuity in the mass flux across an interior boundary or a border between parts in an assembly; it is not applicable to exterior boundaries.

**Axial symmetry** The axial symmetry condition is identical to the insulation/symmetry condition. It is available only for axisymmetric models using cylindrical coordinate systems. Use this boundary condition on the symmetry axis.

Specify the boundary conditions in the **Boundary Settings** dialog box. For boundary conditions on pair boundaries, click the **Pair** tab above the selection area in the left part of the dialog box.



### *Application Mode Variables*

The Diffusion application mode uses the following expressions and coefficients in boundary conditions, equations, and for postprocessing purposes.

NAME	TYPE	DESCRIPTION	EXPRESSION
c	S/B	Concentration	$c$
grad_c, c <sub>x<sub>i</sub></sub>	S/V	Concentration gradient	$ \nabla c , \frac{\partial c}{\partial x_i}$
dflux_c	S	Diffusive flux	$ \underline{D}\nabla c $
ndflux_c	B	Normal diffusive flux	$\mathbf{n} \cdot (-\underline{D}\nabla c)$
dflux_c_x <sub>i</sub>	V	Diffusive flux, $x_i$ component	$\sum_j -D_{ij} \frac{\partial c}{\partial x_j}$
Dts_c	S	Time-scaling coefficient	$\delta_{ts}$
D_c, D <sub>x<sub>i</sub>x<sub>j</sub></sub> _c	S	Diffusion coefficient	$D, D_{ij}$
R_c	S	Reaction rate	$R$
N_c	B	Inward diffusive flux	$N_0$
kc_c	B	Mass-transfer coefficient	$k_c$

NAME	TYPE	DESCRIPTION	EXPRESSION
cb_c	B	Bulk concentration	$c_b$
c0_c	B	Concentration	$c_0$

The vector expressions indicated with a V in the Type column are not present in 1D versions of the diffusion application mode.

---

**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: di), for example, `df1ux_d_di`. (This does not apply to the dependent variable for the concentration.)

---

# The Convection and Diffusion Application Mode

This application mode models the most common type of transport in chemical systems: transport by convection and diffusion. You can simulate transport by convection and diffusion in 1D, 2D, and 3D as well as for axisymmetric systems in 1D and 2D.

The dependent variable in the application mode is the mass concentration,  $c$ .

---

**Note:** The optional Chemical Engineering Module contains an application mode for the convection and diffusion of several species. It also supports modeling using pseudo-2D and pseudo-3D geometries.

---

## *PDE Formulation*

---

The equations for the nonconservative and conservative formulations for a species,  $c$ , are:

$$\delta_{ts} \frac{\partial c}{\partial t} + \nabla \cdot (-D \nabla c) = R - \mathbf{u} \cdot \nabla c \quad \text{nonconservative}$$

$$\delta_{ts} \frac{\partial c}{\partial t} + \nabla \cdot (-D \nabla c + c \mathbf{u}) = R \quad \text{conservative}$$

The nonconservative formulation is the default for advection and diffusion types of equations in COMSOL Multiphysics because the software assumes an incompressible fluid. Thus the term  $c \nabla \cdot \mathbf{u}$  equals zero and gets dropped from the nonconservative formulation. This ensures that no nonphysical source term arises from a flow field where the incompressibility constraint,  $\nabla \cdot \mathbf{u} = 0$ , is not absolutely fulfilled.

For stationary analysis the term with the time derivative gets dropped.

## Subdomain Settings

The various coefficients used in the equations are:

COEFFICIENT	VARIABLE	DESCRIPTION
$\delta_{ts}$	Dts_c	Time-scaling coefficient
$D$	D_c	Diffusion coefficient or tensor
$D_{ij}$	Dxixj_c	Diffusion coefficient tensor, $x_i x_j$ component
$R$	R_c	Reaction rate
$u, v, w$	u_c, v_c, w_c	Velocity in the $x_1, x_2,$ and $x_3$ directions

The **Time-scaling coefficient**,  $\delta_{ts}$ , is normally 1, but you can change the time scale, for example, from seconds to minutes by setting it to 1/60.

For equations in 2D or 3D, pay special attention to the isotropic diffusion coefficient,  $D$ . If you select this coefficient, the application mode expands it to the diagonal of the diffusion coefficient tensor, that is,  $Dxixi$  equals  $D$ .

You specify the equation coefficients in the **Subdomain Settings** dialog box.

### ARTIFICIAL DIFFUSION

To specify and activate artificial diffusion:

- 1 Open the **Subdomain Settings** dialog box.
- 2 Click the **Physics** tab.
- 3 With at least one subdomain selected, click the **Artificial Diffusion** button.

This opens the **Artificial Diffusion** dialog box shown in Figure 4-1.

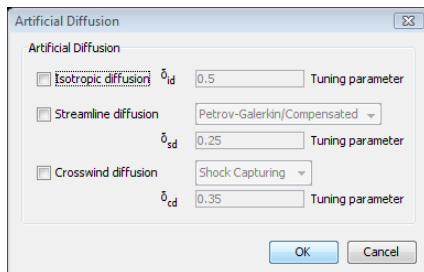


Figure 4-1: The Artificial Diffusion dialog box.

Stabilization is sometimes needed because pure Galerkin discretization is unstable for convectively or source-term dominated transport equations. The stabilization methods



and their effects on the numerical solution are described in Chapter 17, “Stabilization Techniques.”

The Convection and Diffusion application mode supports artificial diffusion using the following methods:

- Isotropic diffusion
- Streamline diffusion
- Crosswind diffusion

#### *Isotropic Diffusion*

Isotropic diffusion is described in the section “Isotropic Diffusion” on page 500.

#### *Streamline Diffusion*

There are three types of streamline diffusion available:

- The *anisotropic diffusion* method is described in the section “Anisotropic Diffusion” on page 501.
- The *Petrov-Galerkin streamline diffusion* method is almost the same as the SUPG method described in the section “Streamline Upwind Petrov-Galerkin (SUPG)” on page 502. The only difference is an intrinsic time scale that the method uses. The convection-diffusion equation uses the original time scale designed for transport equations dominated by convection (see Ref. 1).

There is no strict theory on the value of the tuning parameter,  $\delta_{sd,T}$ , but a good rule of thumb is to select  $\delta_{sd,T} = 0.5/p$  where  $p$  is the order of the basis functions.

- The *Petrov-Galerkin/Compensated* method uses a modified intrinsic time scale that completely eliminates the streamline diffusion contributions in regions dominated by diffusion (Ref. 1).

There is no strict theory on the value of the tuning parameter,  $\delta_{sd,T}$ , but a good rule of thumb is to select  $\delta_{sd,T} = 0.5/p$  where  $p$  is the order of the basis functions.

The drawback is that this formulation can cause discontinuities and that it introduces additional nonlinearities.

### Crosswind Diffusion

Crosswind diffusion provides extra diffusion in the region of sharp gradients. There are two types of crosswind diffusion available:

- **Ordo  $h^{3/2}$** —The Ordo  $h^{3/2}$  crosswind-diffusion algorithm adds a diffusion coefficient according to Ref. 2.

This method should be used only with linear elements but is computational inexpensive. The constant  $\delta_{cd}$  is per default set to 0.35. This value is however only valid for SI units and must be recalculated if other physical units are used.

- **Shock capturing**—The COMSOL Multiphysics implementation adds shock capturing according to:

$$\sum_{e=1}^{N_{el}} \int_{\Omega^e} \frac{1}{2} \alpha_c^e h^e \frac{|\Re(c)|}{|\nabla c|} \nabla \hat{c} \cdot \left( I - \frac{1}{|\mathbf{u}|^2} (\mathbf{u} \otimes \mathbf{u}) \right) \cdot \nabla c d\Omega$$

where  $\Re(c)$  is the equation residual ( $\Re(c) = \nabla \Gamma_c - R$ ) and  $\hat{c}$  is the test function for  $c$ . The effective diffusion coefficient,  $\alpha_c^e$ , is

$$\alpha_c^e = \max(0, \delta_{cd} - 1/\gamma_p^e)$$

$$\gamma_p^e = \frac{|\mathbf{u}| h^e}{2D}$$

$$\beta_p = \frac{R}{|\nabla c|^2} \nabla c$$

where  $D$  is the mean diffusion coefficient and  $R$  is the reaction term. The tuning coefficient  $\delta_{cd}$  is set to 0.35, which correlates to second-order basis functions (see Ref. 3).

### Boundary Conditions

The available boundary conditions are:

BOUNDARY CONDITION	DESCRIPTION
$c = c_0$	Concentration
$-\mathbf{n} \cdot (-D\nabla c + c\mathbf{u}) = N_0$	Flux
$\mathbf{n} \cdot (-D\nabla c + c\mathbf{u}) = 0$	Insulation/Symmetry

BOUNDARY CONDITION	DESCRIPTION
$\mathbf{n} \cdot (-D\nabla c) = 0$	Convective flux
$\mathbf{n}_1 \cdot (-D\nabla c + c\mathbf{u})_1 = \frac{D}{d}(c_1 - c_2)$	Thin boundary layer
$\mathbf{n}_2 \cdot (-D\nabla c + c\mathbf{u})_2 = \frac{D}{d}(c_2 - c_1)$	
$\mathbf{n} \cdot (\mathbf{N}_1 - \mathbf{N}_2) = 0$	Continuity
$-\mathbf{n} \cdot (\mathbf{N}_1 - \mathbf{N}_2) = N_0$	Flux discontinuity
$\mathbf{n} \cdot (-D\nabla c + c\mathbf{u}) = 0$	Axial symmetry

**Concentration** In the equation for the concentration boundary condition,  $c_0$  is a user-specified concentration.

**Flux** In the equation for the flux condition,  $N_0$  is an arbitrary user-specified flux expression.

**Convective flux** This boundary condition typically applies at outlets, where you can assume that mass is transported out of the domain by convection only.

**Thin boundary layer** You can use the thin boundary layer condition to model a thin layer of a material with a small diffusion coefficient compared to the adjacent domains. The layer has the thickness  $d$  and the diffusion coefficient  $D$ . This boundary condition is only available at the border between the parts in an assembly.

**Continuity** This is the default boundary condition on interior boundaries and pair boundaries; it is not applicable to exterior boundaries.

**Flux discontinuity** This boundary condition represents a discontinuity in the mass flux across a border between parts in an assembly.

**Axial symmetry** The axial symmetry condition is identical to the insulation/symmetry condition. It is available only for axisymmetric models using cylindrical coordinate systems. Use this boundary condition only on the symmetry axis.

You specify boundary conditions in the **Boundary Settings** dialog box. For boundary conditions on pair boundaries, click the **Pair** tab above the selection area in the left part of the dialog box.

### *Application Mode Variables*

The Convection and Diffusion application mode uses the following expressions and coefficients in boundary conditions, equations, and for postprocessing purposes:

NAME	TYPE	DESCRIPTION	EXPRESSION
c	S/B	Concentration	$c$
grad_c, $cx_i$	S/V	Concentration gradient	$ \nabla c , \frac{\partial c}{\partial x_i}$
dflux_c	S	Diffusive flux	$ D\nabla c $
cflux_c	S	Convective flux	$ c\mathbf{u} $
tflux_c	S	Total flux	$ -D\nabla c + c\mathbf{u} $
ndflux_c	B	Normal diffusive flux	$\mathbf{n} \cdot (-D\nabla c)$
ncflux_c	B	Normal convective flux	$c\mathbf{n} \cdot \mathbf{u}$
ntflux_c	B	Normal total flux	$\mathbf{n} \cdot (-D\nabla c + c\mathbf{u})$
dflux_c_ $x_i$	V	Diffusive flux, $x_i$ component	$\sum_j -D_{ij} \frac{\partial c}{\partial x_j}$
cflux_c_ $x_i$	V	Convective flux, $x_i$ component	$cu_i$
tflux_c_ $x_i$	V	Total flux, $x_i$ component	$\sum_j -D_{ij} \frac{\partial c}{\partial x_j} + cu_i$
cellPe_c	S	Cell Peclet number	$\left  \frac{\mathbf{u}h}{D} \right $
Dts_c	S	Time-scaling coefficient	$\delta_{ts}$
udl_c	S	Dimensionless velocity	$u_{dl}$
D_c, $D_{xixj\_c}$	S	Diffusion coefficient	$D, D_{ij}$
R_c	S	Reaction rate	$R$
u_c, v_c, w_c	S	Velocity of $c$ , $x_i$ component	$u_i$
N_c	B	Inward flux	$N_0$
c0_c	B	Concentration	$c_0$
beta_c_ $x_i$	S	Convective field, $x_i$ component	$u_i$

NAME	TYPE	DESCRIPTION	EXPRESSION
Dm_c	S	Mean diffusion coefficient	$\frac{\sum_{i,j} D_{ij} \beta_i \beta_j}{ \beta }$
res_c	S	Equation residual	$\nabla \cdot (-D \nabla c + c \mathbf{u}) - R$
res_sc_c	S	Shock capturing residual	$\nabla \cdot (c \mathbf{u}) - R$
da_c	S	Total time-scale factor	$\delta_{ts}$

The vector expressions indicated with a V in the Type column are not present in 1D versions of the Convection and Diffusion application mode.

---

**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: cd), for example, `df1ux_d_cd`. (This does not apply to the dependent variable for the concentration.)

---

### References

1. C. Johnson, *Numerical solution of partial differential equations by the finite element method*, Studentlitteratur, 1987.
2. C. Johnson, A. Schatz, and L. Wahlbin, “Crosswind Smear and Pointwise Error in Streamline Diffusion Finite Element Methods,” *Math. Comp.*, vol. 49, pp. 25–38, 1987.
3. R. Codina, “A Shock-Capturing Anisotropic Diffusion for the Finite Element Solution of the Diffusion-Convection-Reaction Equation,” *VIII International Conference on Finite Elements in Fluids*, vol. 1, pp. 67–75.

# Example—Effective Diffusivity in Porous Materials

This model introduces the concept of effective diffusivity in porous media by comparing the transport through an artificial porous structure described in a detailed model with a simplified homogeneous porous media approach using effective transport properties.

The exercise consists of two parts. The first part describes how to create the model with a detailed geometry. The second part shows how to define a homogeneous model for porous media using an effective diffusivity calculated using the detailed model from the first part.

## *Introduction*

---

Transport through porous structures is usually treated using simplified homogeneous models with effective transport properties. This is in most cases a necessity, since the typical dimensions of the pores and particles making up the porous structure are several orders of magnitude smaller than the size of the subdomain that is to be modeled.

However, it might be interesting to investigate the assumptions and simplifications done when homogenizing a porous structure by comparing a homogeneous model with a model defined using the detailed structure.

The artificial porous structure used in this model is depicted in Figure 4-2 below.

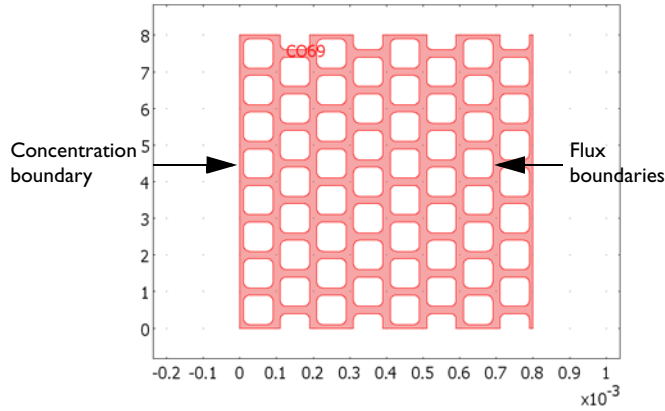


Figure 4-2: Artificial porous structure. The domain colored in red is accessible for diffusion.

### Model Definition

The model equation in the modeled domain shown in Figure 4-2 is the time-dependent equation

$$\frac{\partial c}{\partial t} + \nabla \cdot (-D\nabla c) = 0$$

where  $c$  denotes concentration (mol/m<sup>3</sup> using SI units) and  $D$  the diffusion coefficient (m<sup>2</sup>/s) of the solute.

The boundary conditions are of three different types. A concentration boundary condition applies at the left vertical boundary in Figure 4-2. It is expressed as

$$c = c_0$$

where  $c_0$  is a given concentration.

The right vertical boundary in Figure 4-2 is set according to

$$(-D\nabla c) \cdot \mathbf{n} = k_m(c - c_1)$$

where  $k_m$  is the mass transfer coefficient (m/s), and  $c_1$  is the concentration in a bulk solution outside of the porous structure.

All other boundaries are insulating boundaries according to

$$(-D\nabla c) \cdot \mathbf{n} = 0$$

The initial condition is given by a bell-shaped profile along the  $x$ -axis with its maximum at  $x = 0$  and a corresponding value of  $c = c_0$ :

$$c(t_0) = c_0 \exp(-ax^2)$$

Assume a gaseous solution with a solute content of  $3 \text{ mol/m}^3$  at the concentration boundary. The diffusion coefficient is set to  $1 \cdot 10^{-5} \text{ m}^2/\text{s}$ .

The second part of this exercise uses a homogenized 1D model geometry with effective transport properties and an average porosity. The model equation then becomes:

$$\varepsilon \frac{\partial c}{\partial t} + \nabla \cdot (-D^{\text{eff}} \nabla c) = 0$$

where  $\varepsilon$  denotes the average porosity and  $D^{\text{eff}}$  the effective diffusivity. These properties are calculated from the results of the detailed structure; see the next section. At the boundaries, the concentration and flux conditions described above apply.

### Results and Discussion

The simulations are run for  $t = 0$  to  $0.1 \text{ s}$ , when the simulation reaches steady state. Figure 4-3 below shows the concentration profile after  $0.05 \text{ s}$  in the porous structure. Already at this stage the concentration has almost reached steady state, which is visible by the nearly linear concentration profile across the structure.

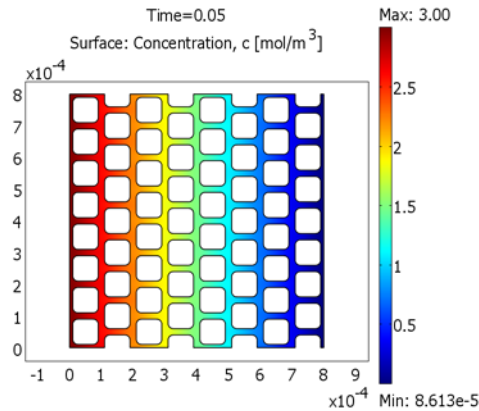


Figure 4-3: Concentration profile in the modeled artificial porous structure at  $t = 0.05 \text{ s}$ .



When modeling porous media, the exact concentration in the pore structure is not the most important issue because the description of the structure is homogenized and not detailed as in Figure 4-3. The most interesting issue is then the description of the flux. To calculate the average flux, integrate over the flux boundary and divide by its length,  $L_0$ , which yields the following expression:

$$N_{\text{average}} = \frac{1}{L_0} \int_0^{L_0} k_m (c - c_1) dS$$

Figure 4-4 shows the value of this integral as a function of time. If you let the process reach steady-state, the average flux becomes  $8.051 \cdot 10^{-3} \text{ mol} \cdot \text{m}^2 / \text{s}$ . Considering the almost linear profile across the structure, it is natural to replace the porous structure with a 1D homogenized structure along the  $x$ -axis. It is then possible to calculate the effective diffusivity according to the following:

$$D^{\text{eff}} \frac{(c_0 - c_{\text{out}})}{L_1} = N_{\text{average}}$$

where  $c_{\text{out}}$  is the average concentration ( $\text{mol}/\text{m}^3$ ) at the flux boundary, and  $L_1$  is the length of the geometry along the  $x$ -axis. The average concentration is obtained by integrating according to the expression below:

$$c_{\text{out}} = \frac{1}{L_0} \int_0^{L_0} c dS.$$

This gives an average concentration  $c_{\text{out}} = 1.61 \cdot 10^{-3} \text{ mol}/\text{m}^3$  and using  $L_1 = 8 \cdot 10^{-4} \text{ m}$  gives the effective diffusivity according to the following:

$$D^{\text{eff}} = \frac{8.051 \cdot 10^{-3} \times 8.0 \cdot 10^{-4}}{(3 - 1.61 \cdot 10^{-3})}$$

which yields a value for the effective diffusivity of  $2.15 \cdot 10^{-6} \text{ m}^2 / \text{s}$  compared to the “free” diffusivity of  $1 \cdot 10^{-5} \text{ m}^2 / \text{s}$ . The effective and “free” diffusivities are usually related according to the equation

$$D^{\text{eff}} = D \frac{\varepsilon}{\tau}$$

where  $\varepsilon$  is the porosity of the structure and  $\tau$  the tortuosity, which is a measure of the actual length per unit effective length a molecule has to diffuse in a porous structure.

To calculate the porosity of the modeled structure, you integrate the value 1 over the structure and then divide this by the length and width of the structure:

$$\varepsilon = \frac{1}{L_0 L_1} \int_0^{L_1} \int_0^{L_0} 1 dx dy$$

resulting in a value of 0.382. The value of  $\tau$  can then be calculated to 1.78. In addition, the tortuosity is usually expressed as a power of the porosity, resulting in an expression for the effective diffusivity according to

$$D^{\text{eff}} = D\varepsilon^p$$

If you use the calculated values for porosity and effective diffusivity, the value for  $p$  is 1.60. The experimental values for  $p$  for porous structures used for transport in catalysts, soils, and other porous structures is usually in the range 1.5–2.

Using the value of the effective diffusivity, a simple homogenized 1D model provides the possibility to compare the value of the flux using a homogenized model to the value using the detailed 2D structure. Figure 4-4 shows that there is an excellent agreement between the model using a detailed geometry and the homogenized model.

The difference in the time-dependent flux is hardly visible between the two cases in the graph.

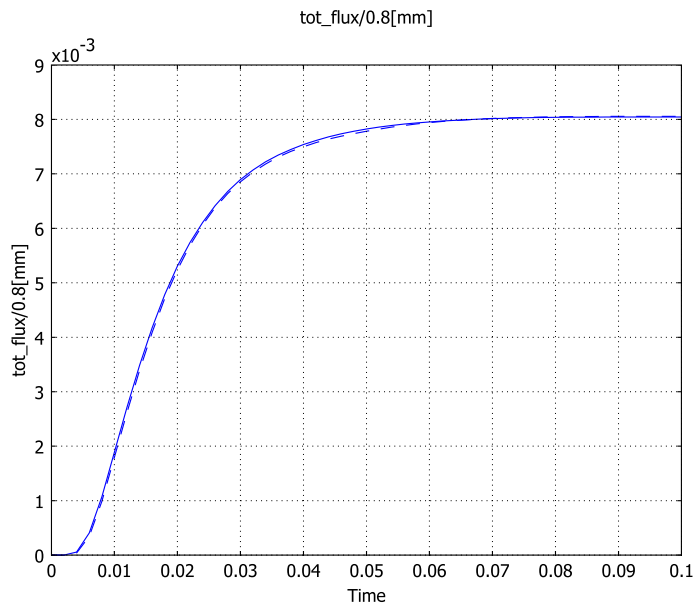


Figure 4-4: Average flux at the flux boundary in the detailed 2D model (solid line) and the 1D homogenized approximation (dashed line).

### Modeling in COMSOL Multiphysics

---

Both models described above are straightforward to define in COMSOL Multiphysics. One feature that is of great use in this model is the ability to define integration coupling variables to automatically generate the values of the integrals needed to evaluate the results of the model. The definition of these integrals is described in detail in the step-by-step instructions below.

---

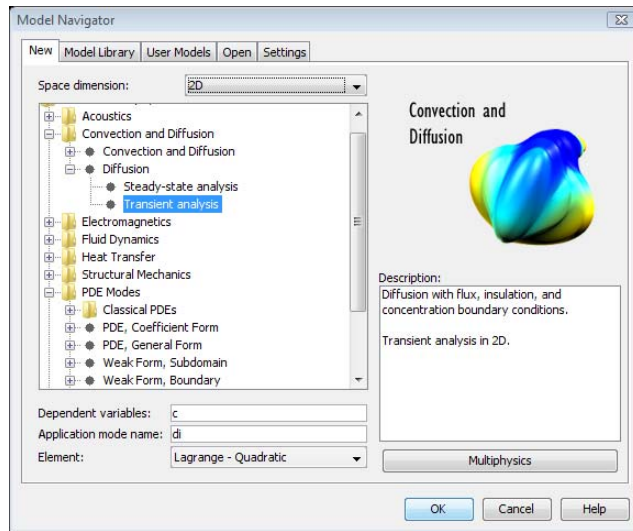
**Model Library path:** COMSOL\_Multiphysics/Diffusion/  
effective\_diffusivity

---

## Modeling Using the Graphical User Interface

You can start by setting up the first and most extensive part of the exercise, which is the 2D model of the detailed porous structure.

- 1 Double-click on the COMSOL Multiphysics icon on your desktop to open the **Model Navigator**.
- 2 Select **2D** from the **Space dimension** list.
- 3 Select **COMSOL Multiphysics>Convection and Diffusion>Diffusion>Transient analysis** as in the figure below.



- 4 Click **OK**.

### OPTIONS AND SETTINGS

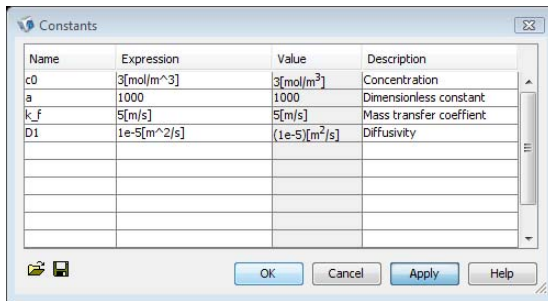
Define the constants required as input to the model.

- 1 Select **Constants** in the **Options** menu.
- 2 Define the concentration  $c_0$  in the **Name** column in the **Constants** dialog box.
- 3 Type  $3[\text{mol}/\text{m}^3]$  in the corresponding **Expression** column.
- 4 Type **Concentration** in the corresponding **Description** column.

The **Name** column defines the name that you have to use to refer back to this constant, the **Expression** column is used to calculate its **Value**, and the **Description** column is used to make notes regarding this constant. Any constant can be a

function of the other constants, which is the reason why the **Expression** column is not always identical to the **Value** column.

- 5 Below the above constant, type a in the **Name** column, 1000 in the **Expression** column, and Dimensionless constant in the **Description** column.
- 6 Continue generating the constant list by typing k\_f in the **Name** column, 5[m/s] in the **Expression** column, and Mass transfer coefficient in the **Description** column.
- 7 Type D1 in the **Name** column, 1e-5[m^2/s] in the **Expression** column, and Diffusivity in the **Description** column. The constant dialog box should look according to the figure below. You can now save this list for later access in the 1D model in the second part of the exercise.



- 8 Click **Apply**.
- 9 Click the **Export Variables To File** button in the **Constants** dialog box (floppy disk symbol).
- 10 Save the data file in your working directory as diffusion.txt.
- 11 Click **Save** in the **Export Variables** dialog box.
- 12 Click **OK** in the **Constants** dialog box.

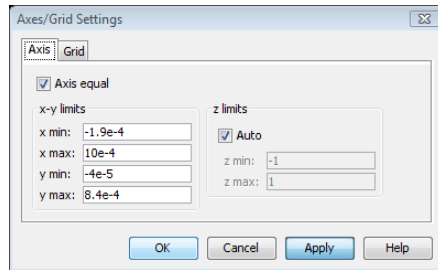
Continue with the functions.

- 1 In the **Options** menu, choose **Expressions>Scalar Expressions**.
- 2 Define a concentration function c1 in the **Name** column in the **Scalar Expressions** dialog box.
- 3 Type  $c0 \cdot \exp(a \cdot (-x/4e-4[m])^2)$  in the corresponding **Expression** column.
- 4 Type Concentration in the corresponding **Description** column.
- 5 Click **Apply**. You can now save the expression for use in the 1D model.

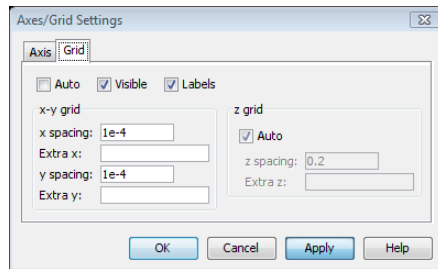
- 6 Click the **Export Variables To File** button in the **Scalar Expressions** dialog box (floppy disk symbol).
- 7 Save the data file in your working directory as `diffusion1.txt`.
- 8 Click **Save** in the **Export Variables** dialog box.
- 9 Click **OK** in the **Scalar Expressions** dialog box.

The last step in the **Options** menu is to set the size of the drawing table.

- 1 Select **Axes/Grid Settings** in the **Options** menu.
- 2 Set the minimum and maximum values for the *x*- and *y*-axes according to the figure below. You can do this by clicking on the edit fields and typing the corresponding values.



- 3 Click the **Grid** tab and clear the **Auto** check box.
- 4 Set the **x spacing** and **y spacing** grid lines as in the figure below.

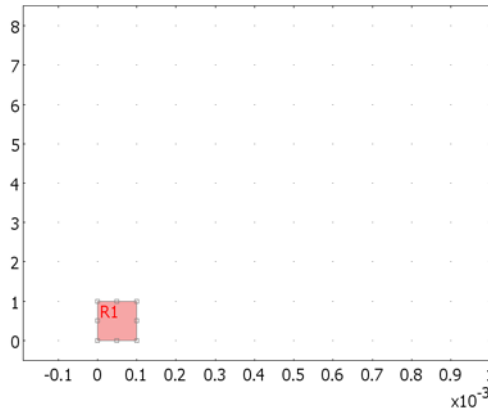


- 5 Click **OK**.

## GEOMETRY MODELING

- 1 Click the **Rectangle/Square** button on the Draw toolbar.

- Use the mouse to drag from the  $(x, y)$  coordinates  $(0, 0)$  to  $(1e-4, 1e-4)$  to create a square with one corner at the origin and one corner across the diagonal from origin at  $(1e-4, 1e-4)$ ; see the figure below.

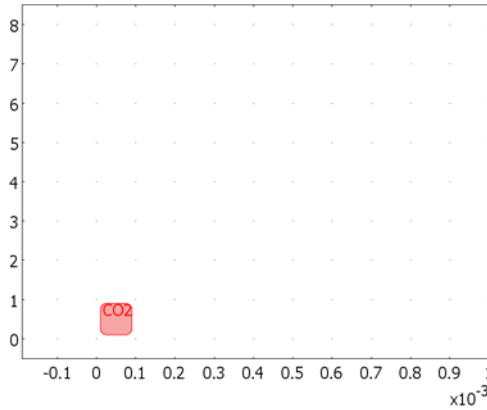


- Click the **Fillet/Chamfer** button on the Draw toolbar.
- Select Vertex 1 in the **Vertex selection** list.
- Type  $2e-5$  in the **Radius** edit field.
- Click **Apply**. The rectangle R1 now changes name to CO1 (composite object 1).
- Select Vertex 2 in the **Vertex selection** list.
- Click **Apply**. The name is now changed to CO2.
- Select Vertex 5 in the **Vertex selection** list.
- Click **Apply**. The name is now changed back to CO1 (this since the name is not taken anymore).
- Select Vertex 7 in the **Vertex selection** list.
- Click **Apply**. The name is now changed again back to CO2.
- Click **OK**.

You should now have the rectangle according to the figure above but now with rounded corners.

- Click the **Scale** button (with CO2 selected; that is, colored red).
- Type  $0.8$  in both the **x** and **y** edit fields in the **Scale factor** area, then click **OK**.
- Click the **Move** button on the Draw toolbar.
- Type  $1e-5$  in both the **x** and **y Displacement** edit fields.

The rounded square should now be positioned according to the figure below.

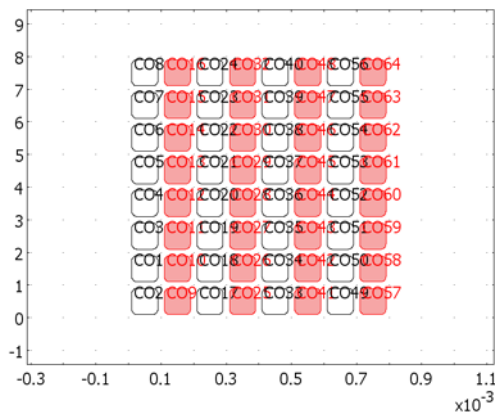


You can now copy and paste to create the porous structure.

- 1 Click the **Array** button on the Draw toolbar.
- 2 Enter  $1e-4$  in the **x** and **y** edit fields in the **Displacement** area.
- 3 In the **Array size** edit fields enter 8 for both **x** and **y**.
- 4 Click **OK**.

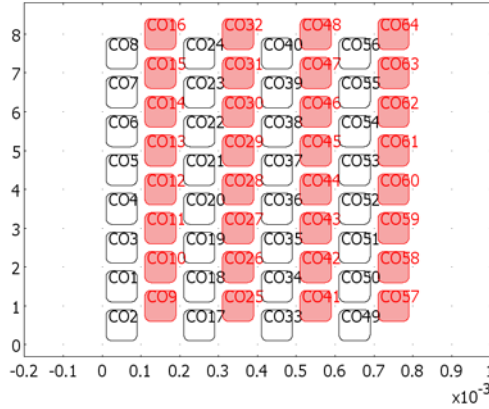
You now have to select each second column of objects and displace them.

- 1 Press the Ctrl key and drag the mouse over the objects in each second column to obtain the selection according to the figure below. You can use the rubber band functionality by dragging the mouse. You keep the selection by pressing the Ctrl key.

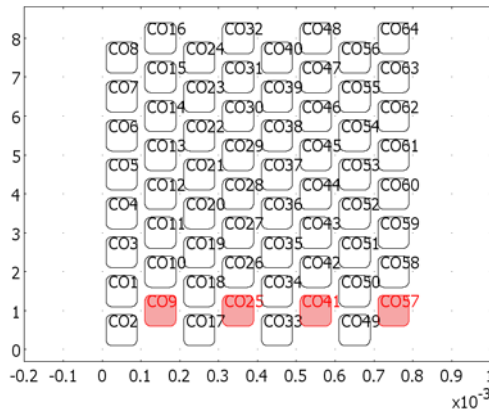




- 2 Click the **Move** button on the Draw toolbar.
- 3 Type  $5e-5$  in the **y** edit field in the **Displacement** area.
- 4 Click **OK**.
- 5 Click **Zoom Extents**. Your graphical user interface should look as in the figure below.

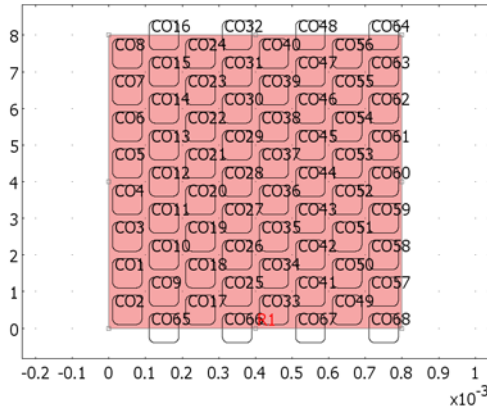


- 6 Choose **Deselect All** from the **Edit** menu.
- 7 Press the Ctrl key and click the objects CO9, CO25, CO41, CO57 to obtain the selection according to the following figure.



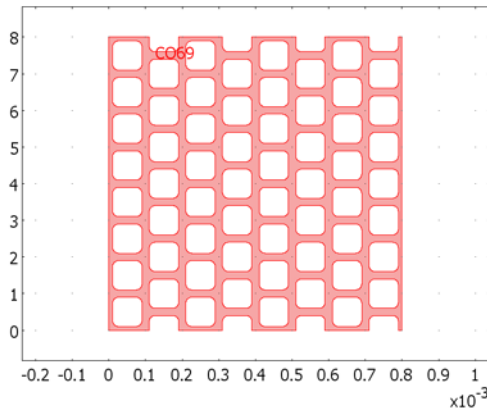
- 8 Press Ctrl+C to copy and Ctrl+V to paste the selected objects.
- 9 In the **Paste** dialog box, type  $-1e-4$  in the **y** edit field in the **Displacement** area. Click **OK**.
- 10 Click the **Zoom Extents** button on the Main toolbar.

- I1** Click the **Rectangle/Square** button and drag the mouse from the coordinates  $(x, y)$   $(0, 0)$  to  $(8 \cdot 10^{-4}, 8 \cdot 10^{-4})$  to create a square with one corner at origin and the other corner across the diagonal from the origin at  $(8 \cdot 10^{-4}, 8 \cdot 10^{-4})$ . The graphical user interface should look according to the figure below.



- I2** Press **Ctrl+A** to select all objects.

- I3** Click the **Difference** button. The geometry should be according to the figure below.



This finalizes the geometry section for the first part of this exercise. You can now continue with the properties and boundary conditions.

## PHYSICS SETTINGS

### Subdomain Settings

- I** Select **Subdomain Settings** from the **Physics** menu.

- 2 Select Subdomain 1 from the **Subdomain selection** list.
- 3 Type D1 in the **Diffusion coefficient** edit field. Note that you have defined D1 in the **Constants** dialog box in the **Options** menu.
- 4 Click the **Init** tab to set the initial condition.
- 5 Type c1 in the **Concentration, c** edit field (**c(t<sub>0</sub>)**). Note that you have defined c1 as a scalar expression variable.
- 6 Click **OK**.

#### *Boundary Conditions*

- 1 Select **Boundary Settings** from the **Physics** menu.
- 2 Select Boundary 1 in the **Boundary selection** list.
- 3 Select **Concentration** from the **Boundary condition** list.
- 4 Type c0 in the **Concentration** edit field.
- 5 Select Boundary 276 from the **Boundary selection** list.
- 6 Select **Flux** from the **Boundary condition** list.
- 7 Type k\_f in the **Mass transfer coefficient** edit field. The bulk concentration in the **Bulk concentration** edit field should be 0.
- 8 Click **OK**.

All other boundaries, except for the two boundaries that you have altered above, should remain at the default Insulation/Symmetry boundary condition.

### **MESH GENERATION**

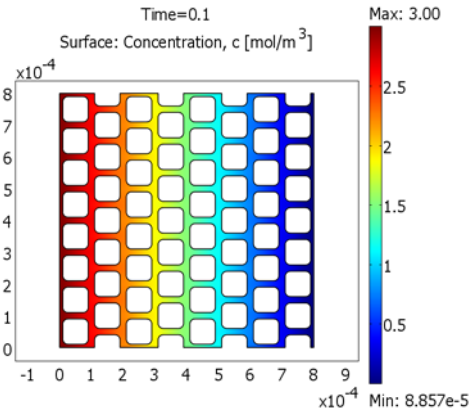
Click the **Initialize Mesh** button on the Main toolbar.

### **COMPUTING THE SOLUTION**

- 1 Click the **Solver Parameters** button on the Main toolbar.
- 2 Type 0:2e-3:0.1 in the **Times** edit field to define the **Time stepping**. This gives an integration from 0 to 0.1 s with 2 ms increments between each output.
- 3 Select **Direct (SPOOLES)** from the **Linear system solver** list. The SPOOLES solver makes use of the symmetry in the diffusion equation, which the solver detects automatically, thus saving memory.
- 4 Click **OK**.
- 5 Click the **Solve** button on the Main toolbar.

## POSTPROCESSING AND VISUALIZATION

The default plot gives the concentration at the final time step 0.1 s; see the figure below.



You can now continue generating the integral expressions needed to evaluate the average flux. To plot the integrals as a function of time, you have to create an integration coupling variable.

- 1 In the **Options** menu, select **Integration Coupling Variables>Boundary Variables**.
- 2 Select Boundary 276 in the **Boundary selection** list.
- 3 Type `tot_flux` in the **Name** column.
- 4 Type `k_f*c` in the **Expression** column.
- 5 Clear the **Global destination** check box.
- 6 Click the **Destination** tab.
- 7 Check the box for Point 532 in the **Point selection** list.
- 8 Click **OK**.

You have now defined the integration variable. However, the value has not been evaluated yet. To do that, you do not have to run the solution process; instead just update the solution:

Select **Update Model** from the **Solve** menu.

This should evaluate the integral at all output times. You can now plot the value of the average flux:

- 1 Select **Domain Plot Parameters** from the **Postprocessing** menu.

- 2 Click the **Point** tab. Select Point 532 in the **Point selection** list.
- 3 Type `tot_flux/0.8[mm]` in the **Expression** edit field to plot the average flux (0.8 mm is the length of the integration boundary).
- 4 Click **OK**.

This generates the solid line in Figure 4-4.

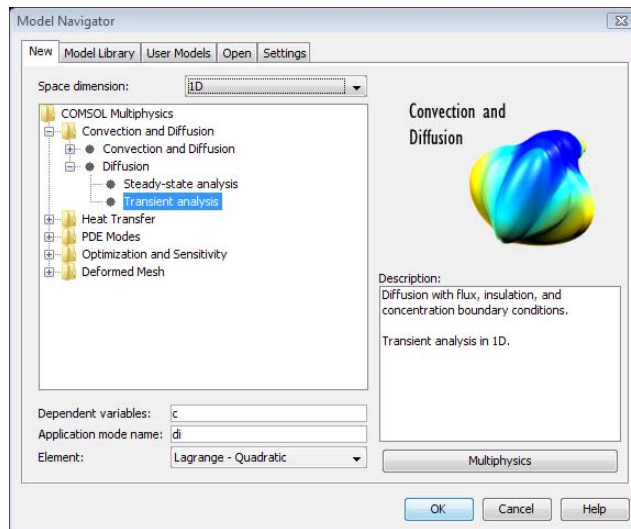
To get the porosity of the domain for the 1D model, do a subdomain integration:

- 1 Select **Subdomain Integration** from the **Postprocessing** menu.
- 2 Select Subdomain 1 in the **Subdomain selection** list.
- 3 Type `1/(0.8[mm])^2` in the **Expression** edit field.
- 4 Click **OK**. The value 0.381976 should be visible in the message window.
- 5 Save the model in your working directory.

You can now continue with the 1D model. Note that the figure window denoted **Figure 1**, showing the average flux of species, should still be open.

#### MODEL NAVIGATOR

- 1 Click the **New** button. Select **ID** from the **Space dimension** list.
- 2 Select **COMSOL Multiphysics>Convection and Diffusion>Diffusion>Transient analysis** from the **Application Modes** tree; see the figure below.



- 3 Click **OK**.

## OPTIONS AND SETTINGS

You can now load the constants from the 2D model above.

- 1 Select **Constants** from the **Options** menu.
- 2 Click the **Import Variables From File** button in the lower left corner of the **Constants** dialog box (open-catalog symbol).
- 3 Select the file `diffusion.txt` that you have previously created in your working directory in the **Import Variables** dialog box.
- 4 Click **Open**.
- 5 Define a new constant in the **Constants** dialog box by typing `epsilon` in the **Name** column, `0.381976` in the **Expression** column, and **Porosity** in the **Description** column.
- 6 Change the diffusivity `D1` to `2.15e-6` in the corresponding **Expression** field.
- 7 Click **OK** in the **Constants** dialog box.

You can continue by loading the scalar expression from the 2D model.

- 1 Select **Expressions>Scalar Expressions** from the **Options** menu.
- 2 Click the **Import Variables From File** button in the lower left corner of the **Scalar Expressions** dialog box (open-catalog symbol).
- 3 Select the file `diffusion1.txt` that you have previously created in your working directory in the **Import Variables** dialog box.
- 4 Click **Open**.
- 5 Click **OK** in the **Scalar Expressions** dialog box.

## GEOMETRY MODELING

- 1 Select **Specify Objects>Line** from the **Draw** menu.
- 2 Type `0 8e-4` in the **Coordinates, x** edit field to create a line from  $x = 0$  to  $x = 8 \cdot 10^{-4}$ .
- 3 Click the **Zoom Extents** button on the Main toolbar.

## PHYSICS

### *Subdomain Settings*

- 1 Select **Subdomain Settings** from the **Physics** menu.
- 2 Select Subdomain 1 from the **Subdomain selection** list.
- 3 Type `epsilon` in the **Time-scaling coefficient** edit field.
- 4 Type `D1` in the **Diffusion coefficient** edit field.

- 5 Click the **Init** tab.
- 6 Type  $c_1$  in the **Concentration, c** edit field ( $c(t_0)$ ). This sets the initial condition.
- 7 Click **OK**.

#### *Boundary Conditions*

- 1 Select **Boundary Settings** from the **Physics** menu.
- 2 Select Boundary 1 from the **Boundary selection** list.
- 3 Select **Concentration** from the **Boundary condition** list.
- 4 Type  $c_0$  in the **Concentration** edit field.
- 5 Select Boundary 2 from the **Boundary selection** list.
- 6 Select **Flux** from the **Boundary condition** list.
- 7 Type  $k_f$  in the **Mass transfer coefficient** edit field.
- 8 Click **OK**.

#### **MESH GENERATION**

Click the **Refine Mesh** button on the Main toolbar to create a fine mesh.

#### **COMPUTING THE SOLUTION**

- 1 Click the **Solver Parameters** button on the Main toolbar.
- 2 Type  $0:2e-3:0.1$  in the **Times** edit field to define the **Time stepping**.
- 3 Click **OK**.
- 4 Click the **Solve** button on the Main toolbar.

#### **POSTPROCESSING AND VISUALIZATION**

- 1 Select **Domain Plot Parameters** from the **Postprocessing** menu.
- 2 Click the **Point** tab. Select Point 2 from the **Point selection** list.
- 3 Type  $k_f*c$  in the **Expression** edit field.
- 4 Click the **Line Settings** button.
- 5 Select **Dashed line** from the **Line style** list, then click **OK**.
- 6 Click the **General** tab. Select the **Keep current plot** check box.
- 7 Select **Figure 1** from the **Plot in** list, then click **OK** to create a plot according to Figure 4-4 on page 63.





# Electromagnetics

This chapter explains the application modes in COMSOL Multiphysics for electromagnetics and how to use them for electromagnetic field simulations. Specifically, it takes a detailed look at the Conductive Media DC application mode, the Electrostatics application mode, the Magnetostatics application mode, and the AC Power Electromagnetics application mode. It concludes by presenting three step-by-step examples using the Conductive Media application mode DC to model a copper plate, the Electrostatics application mode to model an electric sensor, and the Magnetostatics application mode to model a permanent magnet.

# The Electromagnetics Application Modes

For simulating electromagnetic fields, COMSOL Multiphysics offers four application modes.

The first three perform static simulations:

- Electrostatics
- Conductive media DC
- Magnetostatics

With static problems you solve for either electric properties, as is the case for the first two modes, or magnetic properties, as in magnetostatics.

In time-varying electromagnetic fields the electric and magnetic quantities are coupled, so you simulate them with the fourth application modes of interest here: AC Power Electromagnetics.

This section of the manual begins with a brief introduction to electromagnetics and a definition of the electromagnetic quantities. Then it discusses each of the four application modes in detail.

---

**Note:** The optional AC/DC Module contains specialized and extended application modes and models for electromagnetic simulations, for example, for computations of inductors and capacitors. The optional RF Module includes application modes for wave-propagation simulations that are especially useful in microwave engineering and photonics.

---

# Fundamentals of Electromagnetics

The problem of electromagnetic analysis on a macroscopic level is that of solving *Maxwell's equations* subject to certain boundary conditions. Maxwell's equations are a set of equations, written in differential or integral form, stating the relationships between the fundamental electromagnetic quantities. These quantities are:

- The *electric field intensity*,  $\mathbf{E}$
- The *electric displacement* or *electric flux density*,  $\mathbf{D}$
- The *magnetic field intensity*,  $\mathbf{H}$
- The *magnetic flux density*,  $\mathbf{B}$
- The *current density*,  $\mathbf{J}$
- The *electric charge density*,  $\rho$

You can formulate the equations in differential or integral form. This discussion presents them in differential form because it leads to differential equations that the finite element method can handle.

For general time-varying fields, Maxwell's equations are

$$\begin{aligned}\nabla \times \mathbf{H} &= \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} \\ \nabla \times \mathbf{E} &= -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla \cdot \mathbf{D} &= \rho \\ \nabla \cdot \mathbf{B} &= 0\end{aligned}$$

The first two equations are also referred to as *Maxwell-Ampère's law* and *Faraday's law*, respectively. The last two are forms of *Gauss' law* in the electric and magnetic form, respectively.

Another fundamental relationship is the *equation of continuity*:

$$\nabla \cdot \mathbf{J} = -\frac{\partial \rho}{\partial t}$$

Out of these five equations only three are independent. The first two combined with either the electric form of Gauss' law or the equation of continuity form an independent system.

## CONSTITUTIVE RELATIONSHIPS

To obtain a closed system, you need the *constitutive relationships* describing the macroscopic properties of the medium. They are

$$\begin{aligned}\mathbf{D} &= \epsilon_0 \mathbf{E} + \mathbf{P} \\ \mathbf{B} &= \mu_0 (\mathbf{H} + \mathbf{M}) \\ \mathbf{J} &= \sigma \mathbf{E}\end{aligned}\tag{5-1}$$

where  $\epsilon_0$  is the *permittivity of vacuum*,  $\mu_0$  is the *permeability of vacuum*, and  $\sigma$  is the *electric conductivity*. In the SI system the permeability of a vacuum is  $4\pi \cdot 10^{-7}$  H/m. The velocity of an electromagnetic wave in a vacuum is given as  $c_0$ , and you can derive the permittivity of a vacuum from the relationship

$$\epsilon_0 = \frac{1}{c_0^2 \mu_0} = 8.854 \cdot 10^{-12} \text{ F/m} \approx \frac{1}{36\pi} \cdot 10^{-9} \text{ F/m}$$

The *electric polarization vector*  $\mathbf{P}$  describes how a material is polarized when an electric field  $\mathbf{E}$  is present. It can be interpreted as the volume density of electric dipole moments.  $\mathbf{P}$  is generally a function of  $\mathbf{E}$ . Some materials can have a nonzero  $\mathbf{P}$  in the absence of an electric field.

The *magnetization vector*  $\mathbf{M}$  similarly describes how a material is magnetized when a magnetic field  $\mathbf{H}$  is present. It can be interpreted as the volume density of magnetic dipole moments.  $\mathbf{M}$  is generally a function of  $\mathbf{H}$ . One use of the magnetization vector is to describe permanent magnets, which have a nonzero  $\mathbf{M}$  when no magnetic field is present.

For linear materials the polarization is directly proportional to the electric field,  $\mathbf{P} = \epsilon_0 \chi_e \mathbf{E}$ , where  $\chi_e$  is the *electric susceptibility*. Similarly, in linear materials the magnetization is directly proportional to the magnetic field,  $\mathbf{M} = \chi_m \mathbf{H}$ , where  $\chi_m$  is the *magnetic susceptibility*. For such materials the constitutive relations are

$$\begin{aligned}\mathbf{D} &= \epsilon_0 (1 + \chi_e) \mathbf{E} = \epsilon_0 \epsilon_r \mathbf{E} = \epsilon \mathbf{E} \\ \mathbf{B} &= \mu_0 (1 + \chi_m) \mathbf{H} = \mu_0 \mu_r \mathbf{H} = \mu \mathbf{H}\end{aligned}$$

where  $\epsilon_r$  is the material's *relative permittivity*, and  $\mu_r$  is its *relative permeability*. Usually these are scalar properties but can, in the general case, be 3-by-3 tensors when the material is anisotropic. The properties  $\epsilon$  and  $\mu$  (without subscripts) are the material's *permittivity* and *permeability*.

### Generalized Constitutive Relationships

For nonlinear materials, a generalized form of the constitutive relationships is useful. The relationship used for electric fields is

$$\mathbf{D} = \epsilon_0 \epsilon_r \mathbf{E} + \mathbf{D}_r$$

The field  $\mathbf{D}_r$  is the *remanent displacement*, which is the displacement when no electric field is present.

Similarly, a generalized form of the constitutive relationship for the magnetic field is

$$\mathbf{B} = \mu_0 \mu_r \mathbf{H} + \mathbf{B}_r$$

where  $\mathbf{B}_r$  is the *remanent magnetic flux density*, which is the magnetic flux density when no magnetic field is present.

You can generalize the third line in Equation 5-1 by introducing an externally generated current  $\mathbf{J}^e$ . This relationship is then

$$\mathbf{J} = \sigma \mathbf{E} + \mathbf{J}^e$$

### POTENTIALS

Under certain circumstances it can be helpful to formulate a problem in terms of the *electric scalar potential*  $V$  and *magnetic vector potential*  $\mathbf{A}$ . They are given by the equalities

$$\begin{aligned}\mathbf{B} &= \nabla \times \mathbf{A} \\ \mathbf{E} &= -\nabla V - \frac{\partial \mathbf{A}}{\partial t}\end{aligned}$$

which are direct consequences of the magnetic case of Gauss' law and Faraday's law, respectively.

### MATERIAL PROPERTIES

This discussion has so far only formally introduced the constitutive relationships. These seemingly simple relationships can be quite complicated at times. In fact, these relationships require some special considerations when working with four main groups of materials:

- Inhomogeneous materials
- Anisotropic materials

- Nonlinear materials
- Dispersive materials

A material can belong to one or more of these groups.

Inhomogeneous materials are the least complicated. An inhomogeneous medium is one in which the constitutive parameters vary with the space coordinates so that different field properties prevail at different parts of the material structure.

For anisotropic materials the field relationships at any point differ for different directions of propagation. This means that a 3-by-3 tensor is necessary to properly define the constitutive relationships. If this tensor is symmetric, the material is often referred to as *reciprocal*. In these cases you can rotate the coordinate system such that a diagonal matrix results. If two of the diagonal entries are equal, the material is *uniaxially anisotropic*; if none of the elements have the same value, the material is *biaxially anisotropic* (Ref. 2). You need anisotropic parameters, for instance, to examine permittivity in crystals (Ref. 2) and when working with conductivity in solenoids.

In some nonlinear materials the permittivity or permeability depend on the intensity of the electromagnetic field. Nonlinearity also includes hysteresis effects where not only the existing field intensities influence a material's physical properties but the history of the field distribution also plays a role.

Finally, dispersion describes changes in a wave's velocity with wavelength. In the frequency domain you can express dispersion with a frequency dependence of the constitutive laws.

### BOUNDARY AND INTERFACE CONDITIONS

To get a full description of an electromagnetic problem, you must also specify boundary conditions at material interfaces and physical boundaries. At interfaces between two media, you can mathematically express the boundary conditions as

$$\mathbf{n}_2 \times (\mathbf{E}_1 - \mathbf{E}_2) = \mathbf{0}$$

$$\mathbf{n}_2 \cdot (\mathbf{D}_1 - \mathbf{D}_2) = \rho_s$$

$$\mathbf{n}_2 \times (\mathbf{H}_1 - \mathbf{H}_2) = \mathbf{J}_s$$

$$\mathbf{n}_2 \cdot (\mathbf{B}_1 - \mathbf{B}_2) = 0$$

where  $\rho_s$  and  $\mathbf{J}_s$  denote the *surface charge density* and *surface current density*, respectively, and  $\mathbf{n}_2$  is the outward normal from medium 2. Of these four equations,

only two are independent. This is an overdetermined system of equations, so you would like to reduce it. First select either equation one or equation four. Then select either equation two or equation three. Together these selections form a set of two independent conditions.

From these relationships, you can derive the interface condition for the current density,

$$\mathbf{n}_2 \cdot (\mathbf{J}_1 - \mathbf{J}_2) = -\frac{\partial \rho_s}{\partial t}$$

#### *Interface Between a Dielectric and a Perfect Conductor*

A perfect conductor has infinite electrical conductivity and as such has no internal electric field. Otherwise it would produce an infinite current density according to the third fundamental constitutive relationship. At an interface between a dielectric and a perfect conductor, the boundary conditions for the  $\mathbf{E}$  and  $\mathbf{D}$  fields are simplified.

Assume that subscript 1 corresponds to a perfect conductor; then  $\mathbf{D}_1 = \mathbf{0}$  and  $\mathbf{E}_1 = \mathbf{0}$  in the relationships just given. If, in addition, you are dealing with a time-varying case, then  $\mathbf{B}_1 = \mathbf{0}$  and  $\mathbf{H}_1 = \mathbf{0}$ , as well, as a consequence of Maxwell's equations. The result is the following set of boundary conditions for the fields in the dielectric medium for the time-varying case:

$$-\mathbf{n}_2 \times \mathbf{E}_2 = 0$$

$$-\mathbf{n}_2 \times \mathbf{H}_2 = \mathbf{J}_s$$

$$-\mathbf{n}_2 \cdot \mathbf{D}_2 = \rho_s$$

$$-\mathbf{n}_2 \cdot \mathbf{B}_2 = 0$$

#### *Electromagnetic Force for Particle Tracing*

---

The electromagnetic force is available for particle tracing plots in the Conductive Media DC application mode and the Electrostatics application mode in the COMSOL Multiphysics products.

The following expression is used to describe the electromagnetic force on a particle with the charge  $q$  moving through the electric and magnetic fields with the velocity  $\mathbf{v}$ :

$$\mathbf{F} = q(\mathbf{E} + \mathbf{v} \times \mathbf{B})$$

#### **USING THE ELECTROMAGNETIC FORCE FOR PARTICLE TRACING**

The electromagnetic force is the default selection in the **Predefined forces** list on the **Particle Tracing** page.

There is one parameter in this force expression: the particle charge,  $q$ , which you define by clicking the **Parameters** button. The default value of  $q$  is the elementary charge:

PARAMETER NAME IN EQUATION	DESCRIPTION	DEFAULT VALUE	DEFAULT VARIABLE NAME
$q$	particle charge	$1.602 \cdot 10^{-19} \text{ C}$	partq

The particle mass appears in the force equation, and you enter its value in the **Mass** edit field. The default value for the particle mass is  $m_p = 9.1095 \cdot 10^{-31} \text{ kg}$ , which is the mass of an electron.

---

**Note:** The default settings work for models that use SI units.

---

### *Electromagnetic Forces*

---

The electromagnetics application modes that solve for the magnetic vector potential **A** (the Magnetostatics and AC Power Electromagnetics application modes) contain a predefined subdomain variable for computing the Lorentz force, which gives the force distribution exerted on a current-carrying conductor placed in magnetic flux density **B**. The Lorentz force is defined as

$$\mathbf{F} = \mathbf{J} \times \mathbf{B} \quad (5-2)$$

The Lorentz force gives very good accuracy for electromagnetic force calculations in conducting domains. For nonconducting domains you can use a more general method: integrating the Maxwell stress tensor variables over the boundaries of the object for which to calculate the total force. The Maxwell surface stress tensor is available as a boundary variable.

For a full list of variables available within an application mode (including the force variables), see the table at the end of the corresponding application mode section.



# The Conductive Media DC Application Mode

Electrolysis and the computation of resistances of grounding plates involves a medium with electric conductivity  $\sigma$  and a steady current.

You can use the *Conductive Media DC* application mode for 3D, 2D in-plane, and 2D axisymmetric models.

## *PDE Formulation*

---

When handling conductive media you must consider the equation of continuity. In a stationary coordinate system, the point form of *Ohm's law* states that

$$\mathbf{J} = \sigma \mathbf{E} + \mathbf{J}^e$$

where  $\mathbf{J}^e$  is an externally generated current density. The static form of the equation of continuity then gives

$$\nabla \cdot \mathbf{J} = -\nabla \cdot (\sigma \nabla V - \mathbf{J}^e) = 0$$

To handle current sources, generalize the equation to

$$-\nabla \cdot (\sigma \nabla V - \mathbf{J}^e) = Q_j$$

The in-plane *Conductive Media DC* application mode assumes that your model has a symmetry where the electric potential varies only in the  $x$  and  $y$  directions and is constant in the  $z$  direction. This implies that the electric field,  $\mathbf{E}$ , is tangential to the  $xy$ -plane. The application mode solves the following equation where  $d$  is the thickness in the  $z$  direction:

$$-\nabla \cdot d(\sigma \nabla V - \mathbf{J}^e) = dQ_j$$

The axisymmetric *Conductive Media DC* application mode considers the situation where the fields and geometry are axially symmetric. In this case the electric potential is constant in the  $\phi$  direction, which implies that the electric field is tangential to the  $rz$ -plane.

Writing the generalized equation for  $Q_j$  in cylindrical coordinates and multiplying it by  $r$  to avoid singularities at  $r = 0$  results in

$$-\begin{bmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial z} \end{bmatrix}^T \cdot \left( r\sigma \begin{bmatrix} \frac{\partial V}{\partial r} \\ \frac{\partial V}{\partial z} \end{bmatrix} - r\mathbf{J}^e \right) = rQ_j$$

### SPECIFYING THE CONDUCTIVITY

You can provide the conductivity in three different ways:

- Isotropic conductivity: a scalar number or expression
- Anisotropic conductivity: several components of a conductivity tensor to define an anisotropic material
- Temperature-dependent conductivity (which occurs in, for example, Joule heating, which is also called resistive heating). In this case the following equation describes the conductivity:

$$\sigma = \frac{1}{\rho_0(1 + \alpha(T - T_0))}$$

where  $\rho_0$  is the resistivity at the reference temperature  $T_0$ .  $\alpha$  is the temperature coefficient of resistivity, which describes how the resistivity varies with temperature.  $T$  is the current temperature, which can be a value that you specify or the temperature from a heat transfer application mode (in the Joule Heating predefined multiphysics coupling, this is the default setting).

To specify an isotropic or anisotropic conductivity, select **Conductivity** from the **Conductivity relation** list. To use the linear temperature dependence in the equation above, select **Linear temperature relation** from the **Conductivity relation** list. If you want to use another expression for a temperature-dependent conductivity, select **Conductivity** from the **Conductivity relation** list and specify the conductivity as a function of temperature.

### *Boundary Conditions*

---

The relevant interface condition at interfaces between different media for this application mode is

$$\mathbf{n}_2 \cdot (\mathbf{J}_1 - \mathbf{J}_2) = 0$$

This is fulfilled by the natural boundary condition

$$\mathbf{n} \cdot [(\sigma \nabla V - \mathbf{J}^e)_1 - (\sigma \nabla V - \mathbf{J}^e)_2] = -\mathbf{n} \cdot (\mathbf{J}_1 - \mathbf{J}_2) = 0$$

### CURRENT FLOW

The current-flow boundary condition

$$\mathbf{n} \cdot \mathbf{J} = \mathbf{n} \cdot \mathbf{J}_0$$

specifies the normal component of the current density for the current flowing across the boundary.

### INWARD CURRENT FLOW

The inward-current flow boundary condition

$$-\mathbf{n} \cdot \mathbf{J} = J_n$$

is similar to the previous current-flow boundary condition except this case specifies the normal component of the current density rather than the complete vector. When the normal component  $J_n$  is positive then the current flows inwards towards the boundary.

### DISTRIBUTED RESISTANCE

You can use the distributed resistance boundary condition

$$\mathbf{n} \cdot \mathbf{J} = \frac{\sigma}{d}(V - V_{\text{ref}}), \quad \mathbf{n} \cdot (\mathbf{J}_1 - \mathbf{J}_2) = \frac{\sigma}{d}(V - V_{\text{ref}})$$

to model a thin sheet of a resistive material. The sheet has the thickness  $d$  and is connected to the potential  $V_{\text{ref}}$ .

### ELECTRIC INSULATION

The electric-insulation boundary condition

$$\mathbf{n} \cdot \mathbf{J} = 0$$

specifies that no current flows across the boundary.

This boundary condition is also applicable at symmetric boundaries where the potential is known to be symmetric with respect to the boundary.

### ELECTRIC POTENTIAL

The electric-potential boundary condition

$$V = V_0$$

specifies the voltage at a boundary. Because you are solving for the potential in this application mode, you generally define the value of the potential at some boundary in the geometry.

### **GROUND**

The ground boundary condition

$$V = 0$$

is a special case of the previous one but specifies zero potential. This boundary condition is also applicable at symmetry boundaries where the potential is known to be antisymmetric with respect to the boundary.

### **CURRENT SOURCE**

You can apply the current-source boundary condition

$$\mathbf{n} \cdot (\mathbf{J}_1 - \mathbf{J}_2) = J_n$$

to interior boundaries that represent either a current source or sink.

### **CONTINUITY**

The continuity boundary condition

$$\mathbf{n} \cdot (\mathbf{J}_1 - \mathbf{J}_2) = 0$$

specifies that the normal components of the electric current are continuous across the interior boundary.

### **CONTACT RESISTANCE**

You can use the contact resistance boundary condition

$$\mathbf{n} \cdot \mathbf{J}_1 = \frac{\sigma}{d}(V_1 - V_2)$$

$$\mathbf{n} \cdot \mathbf{J}_2 = \frac{\sigma}{d}(V_2 - V_1)$$

to model a thin layer of a resistive material. The layer has the thickness  $d$  and the conductivity  $\sigma$ . This boundary condition is only available at the border between the parts in an assembly.

## AXIAL SYMMETRY

In axisymmetric models, use the boundary condition for axial symmetry at the symmetry axis  $r = 0$ .

### *Line Sources*

---

In 3D you can specify line sources along the edges of a geometry.

#### LINE CURRENT SOURCE

You can apply a line current source  $Q_{jl}$  to edges. This source represents electric current per unit length.

### *Point Sources*

---

It is possible to add point sources to both 2D and 3D models.

#### POINT CURRENT SOURCE

You can apply a point current source  $Q_{j0}$  to points. This source represents an electric current flowing out of the point.

### *Application Mode Variables*

---

The following table shows the fundamental fields, all derivable from the electric potential, that are available for postprocessing and for use in equations and boundary conditions.

NAME	TYPE	DESCRIPTION	EXPRESSION
V	S	electric potential	$V$
sigma	S	electric conductivity	$\sigma$
sigma <sub>ij</sub>	S	electric conductivity, $x_i x_j$ component	$\sigma_{ij}$
Q <sub>j</sub>	S	current source	$Q_j$
J <sub>ei</sub>	S	external current density, $x_i$ component	$J_i^e$
normJ <sub>e</sub>	S	external current density, norm	$\sqrt{\mathbf{J}^e \cdot \mathbf{J}^e}$
J <sub>ii</sub>	S	potential current density, $x_i$ component	$\sigma_{ij} E_j$
normJ <sub>i</sub>	S	potential current density, norm	$\sqrt{\mathbf{J}^i \cdot \mathbf{J}^i}$

NAME	TYPE	DESCRIPTION	EXPRESSION
$J_i$	S	total current density, $x_i$ component	$J_i^e + J_i^i$
normJ	S	total current density, norm	$\sqrt{\mathbf{J} \cdot \mathbf{J}}$
$E_i$	S	electric field, $x_i$ component	$\frac{\partial V}{\partial x_i}$
normE	S	electric field, norm	$\sqrt{\mathbf{E} \cdot \mathbf{E}}$
Q	S	resistive heating	$\mathbf{J} \cdot \mathbf{E}$
nJ	B	current density outflow	$\mathbf{n} \cdot \mathbf{J}$
Qj1	E	line current source	$Q_{jl}$
Qj0	P	point current source	$Q_{j0}$

---

**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: dc), for example, normE\_dc. (This does not apply to the dependent variable for the potential.)

---

# The Electrostatics Application Mode

Applications involving *electrostatics* include high-voltage apparatus, electronic devices, and capacitors. The term “statics” means that the time rate of change of the electric field is slow, and that wavelengths are very large compared to the size of the domain of interest.

The *Electrostatics* application mode is available for 3D, 2D in-plane, and 2D axisymmetric models.

## *PDE Formulation*

---

COMSOL Multiphysics carries out the modeling of static electric fields using the electric potential  $V$ . By combining the definition of potential with Gauss’ law and the equation of continuity, it is possible to derive the classic Poisson’s equation.

Specifically, under static conditions the electric potential,  $V$ , is defined by the relationship

$$\mathbf{E} = -\nabla V$$

Combining this equation with the constitutive relationship  $\mathbf{D} = \epsilon_0 \mathbf{E} + \mathbf{P}$  between  $\mathbf{D}$  and  $\mathbf{E}$ , it is possible to represent Gauss’ law as Poisson’s equation

$$-\nabla \cdot (\epsilon_0 \nabla V - \mathbf{P}) = \rho$$

The In-plane Electrostatics application mode assumes a symmetry where the electric potential varies only in the  $x$  and  $y$  directions and is constant in the  $z$  direction. This implies that the electric field,  $\mathbf{E}$ , is tangential to the  $xy$ -plane. Given this symmetry, you solve the same equation as in the 3D case.

The Axisymmetric Electrostatics application mode considers the situation where the fields and geometry are axially symmetric. In this case the electric potential is constant in the  $\varphi$  direction, which implies that the electric field is tangential to the  $rz$ -plane.

Writing the previous equation for  $\rho$  in cylindrical coordinates and multiplying it by  $r$  to avoid singularities at  $r = 0$ , the equation becomes

$$-\begin{bmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial z} \end{bmatrix}^T \cdot \left( r\epsilon_0 \begin{bmatrix} \frac{\partial V}{\partial r} \\ \frac{\partial V}{\partial z} \end{bmatrix} - r\mathbf{P} \right) = r\rho$$

### *Application Scalar Variables*

There is one application-specific scalar variable in this mode:

PROPERTY	NAME	DEFAULT	DESCRIPTION
$\epsilon_0$	epsilon0	$8.854187817 \cdot 10^{-12}$ F/m	Permittivity of vacuum

### *Boundary Conditions*

The relevant interface condition at interfaces between different media for this mode is

$$\mathbf{n}_2 \cdot (\mathbf{D}_1 - \mathbf{D}_2) = \rho_s$$

In the absence of surface charges, this condition is fulfilled by the natural boundary condition

$$\mathbf{n} \cdot [(\epsilon_0 \nabla V - \mathbf{P})_1 - (\epsilon_0 \nabla V - \mathbf{P})_2] = -\mathbf{n} \cdot (\mathbf{D}_1 - \mathbf{D}_2) = 0$$

### **ELECTRIC DISPLACEMENT**

The electric-displacement boundary condition

$$\mathbf{n} \cdot \mathbf{D} = \mathbf{n} \cdot \mathbf{D}_0$$

specifies the normal component of the electric displacement at a boundary.

### **SURFACE CHARGE**

The surface-charge boundary condition

$$-\mathbf{n} \cdot \mathbf{D} = \rho_s, \quad \mathbf{n} \cdot (\mathbf{D}_1 - \mathbf{D}_2) = \rho_s$$

specifies the surface charge density at an outer boundary or at an interior boundary between two nonconducting media.

### **ZERO CHARGE/SYMMETRY**

The zero charge/symmetry boundary condition



$$\mathbf{n} \cdot \mathbf{D} = 0$$

specifies that the normal component of the electric displacement equals zero.

This boundary condition is also applicable at symmetry boundaries where the potential is known to be symmetric with respect to the boundary.

### **ELECTRIC POTENTIAL**

The electric-potential boundary condition

$$V = V_0$$

specifies the voltage at a boundary. Because you are solving for the potential in this application mode, you generally define the value of the potential at some boundary in the geometry.

### **GROUND**

The ground boundary condition

$$V = 0$$

is a special case of the previous one but specifying zero potential. This boundary condition is also applicable at symmetry boundaries where the potential is known to be antisymmetric with respect to the boundary.

### **CONTINUITY**

The continuity boundary condition

$$\mathbf{n} \cdot (\mathbf{D}_1 - \mathbf{D}_2) = 0$$

specifies that the normal component of the electric displacement is continuous across the boundary.

### **THIN LOW PERMITTIVITY GAP**

You can use the thin low permittivity gap condition

$$\mathbf{n} \cdot \mathbf{D}_1 = \frac{\epsilon}{d}(V_1 - V_2)$$

$$\mathbf{n} \cdot \mathbf{D}_2 = \frac{\epsilon}{d}(V_2 - V_1)$$

to model a thin gap of a material with a small permittivity compared to the adjacent domains. The layer has the thickness  $d$  and the relative permittivity  $\epsilon_r$ . This boundary condition is only available at the border between the parts in an assembly.

### AXIAL SYMMETRY

In axisymmetric models, use the boundary condition for axial symmetry on the symmetry axis  $r = 0$ .

### *Line Sources*

---

In 3D models you can specify line sources along the edges of a geometry.

### LINE CHARGE

You can apply a line charge  $Q_l$  along edges. Provide the source as the electric charge per unit length.

### *Point Sources*

---

It is possible to use point sources in 2D and 3D models.

### POINT CHARGE

You can apply a point charge  $Q_0$  at points in a model.

### *Application Mode Variables*

---

The fundamental fields, which are derivable from the electric potential, are available for postprocessing and for use in equations and boundary conditions. The expressions for some variables in the following table vary depending on the constitutive relationship as indicated in the Const. Rel. column; the abbreviations indicate the constitutive relationship as given in the second table.

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
V	S		electric potential	V
epsilon0	S		permittivity of a vacuum	$\epsilon_0$
epsilon <sub>r</sub>	S	epsr, Dr	relative permittivity	$\epsilon_r$
epsilon <sub>r</sub>	S	P	relative permittivity	1
epsilon <sub>rij</sub>	S	epsr, Dr	relative permittivity, $x_i x_j$ component	$\epsilon_{rij}$
epsilon <sub>rij</sub>	S	P	relative permittivity, $x_i x_j$ component	1

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
epsilon	S		permittivity	$\epsilon_0 \epsilon_r$
epsilon <sub>ij</sub>	S		permittivity, $x_i x_j$ component	$\epsilon_0 \epsilon_{rij}$
P <sub>i</sub>	S	P	electric polarization, $x_i$ component	$P_i$
P <sub>i</sub>	S	epsr, Dr	electric polarization, $x_i$ component	$D_i - \epsilon_0 E_i$
normP	S		electric polarization, norm	$\sqrt{\mathbf{P} \cdot \mathbf{P}}$
Dr <sub>i</sub>	S	epsr	remanent displacement, $x_i$ component	0
Dr <sub>i</sub>	S	P	remanent displacement, $x_i$ component	$P_i$
Dr <sub>i</sub>	S	Dr	remanent displacement, $x_i$ component	$D_{ri}$
normDr	S		remanent displacement, norm	$\sqrt{\mathbf{D}_r \cdot \mathbf{D}_r}$
rho	S		space-charge density	$\rho$
E <sub>i</sub>	S		electric field, $x_i$ component	$-\frac{\partial V}{\partial x_i}$
normE	S		electric field, norm	$\sqrt{\mathbf{E} \cdot \mathbf{E}}$
D <sub>i</sub>	S	epsr	electric displacement, $x_i$ component	$\epsilon_{ij} E_j$
D <sub>i</sub>	S	P	electric displacement, $x_i$ component	$\epsilon_0 E_i + P_i$
D <sub>i</sub>	S	Dr	electric displacement, $x_i$ component	$\epsilon_0 \epsilon_{rij} E_j + D_{ri}$
normD	S		electric displacement, norm	$\sqrt{\mathbf{D} \cdot \mathbf{D}}$
We	S		electric-energy density	$\frac{\mathbf{E} \cdot \mathbf{D}}{2}$
nD	B		surface-charge density	$-\mathbf{n} \cdot \mathbf{D}$
Q1	E		line-charge density	$Q_1$
Q0	P		charge	$Q_0$

---

**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: `es`), for example, `normE_es`. (This does not apply to the dependent variable for the potential.)

---

TABLE 5-1: ABBREVIATIONS FOR THE CONSTITUTIVE RELATIONSHIPS

ABBREVIATION	CONSTITUTIVE RELATIONSHIP
<code>epsr</code>	$\mathbf{D} = \epsilon_0 \epsilon_r \mathbf{E}$
<code>P</code>	$\mathbf{D} = \epsilon_0 \mathbf{E} + \mathbf{P}$
<code>Dr</code>	$\mathbf{D} = \epsilon_0 \epsilon_r \mathbf{E} + \mathbf{D}_r$

# Magnetostatics Application Mode

It is often possible to model magnets, electric motors, and transformers with *magnetostatics*. The term “statics” implies that the time rate of change of the magnetic field is slow.

The *Magnetostatics* application mode is available for 2D in-plane and 2D axisymmetric models.

## *PDE Formulation*

---

To derive the equation system this mode solves, start with Ampère’s law for static cases,

$$\nabla \times \mathbf{H} = \mathbf{J}$$

The current is

$$\mathbf{J} = \sigma \mathbf{v} \times \mathbf{B} + \mathbf{J}^e$$

where  $\mathbf{J}^e$  is an externally generated current density, and  $\mathbf{v}$  is the velocity of the conductor.

Using the definitions of magnetic potential,

$$\mathbf{B} = \nabla \times \mathbf{A}$$

and the constitutive relationship,  $\mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M})$ , you can rewrite Ampère’s law as

$$\nabla \times (\mu_0^{-1} \nabla \times \mathbf{A} - \mathbf{M}) - \sigma \mathbf{v} \times (\nabla \times \mathbf{A}) = \mathbf{J}^e$$

In the 2D case there are no variations in the  $z$  direction, and the current is parallel to the  $z$ -axis. Therefore you can add a term  $-\sigma \Delta V / L$  to the definition of the current where  $\Delta V$  is the potential difference over the distance  $L$ . This leads to

$$\mathbf{J} = \sigma \mathbf{v} \times \mathbf{B} + \mathbf{J}^e - \sigma \frac{\Delta V}{L}$$

In 2D, that equation simplifies

$$-\nabla \cdot \left( \mu_0^{-1} \nabla A_z - \begin{bmatrix} -M_y \\ M_x \end{bmatrix} \right) + \sigma \mathbf{v} \cdot \nabla A_z = \sigma \frac{\Delta V}{L} + J_z^e$$

The axisymmetric case uses another form of the contribution of the current coming from a potential difference  $-\sigma(V_{\text{loop}}/(2\pi r))$  because current is present only in the azimuthal direction. The above equation then becomes, in cylindrical coordinates,

$$-\begin{bmatrix} \frac{\partial}{\partial r} \\ \frac{\partial}{\partial z} \end{bmatrix}^T \left( r\mu_0^{-1} \begin{bmatrix} \frac{\partial u}{\partial r} \\ \frac{\partial u}{\partial z} \end{bmatrix} + \mu_0^{-1} \begin{bmatrix} 2 \\ 0 \end{bmatrix} u - \begin{bmatrix} M_z \\ -M_r \end{bmatrix} \right) + r\sigma \begin{bmatrix} \frac{\partial u}{\partial r} \\ \frac{\partial u}{\partial z} \end{bmatrix} + 2\sigma v_r u = \sigma \frac{V_{\text{loop}}}{2\pi r} + J_\varphi^e$$

The dependent variable  $u$  is the nonzero component of the magnetic potential divided by the radial coordinate  $r$ , that is,

$$u = \frac{A_\varphi}{r}$$

The application mode performs this transformation to avoid singularities at the symmetry axis.

### *Application Mode Property*

The application mode property appears in the following table:

PROPERTY	VALUES	DESCRIPTION
Analysis type	Static   Time-harmonic	Specifies which type of analysis to perform

### *Application Scalar Variable*

The application scalar variable in this application mode is the permeability of vacuum.

PROPERTY	NAME	VALUE	DESCRIPTION
$\mu_0$	mu0	$4\pi \cdot 10^{-7}$ H/m	Permeability of vacuum

### *Boundary and Interface Conditions*

The relevant interface conditions are

$$\mathbf{n}_2 \times (\mathbf{H}_1 - \mathbf{H}_2) = \mathbf{J}_s$$

The natural boundary condition fulfills this equation if the surface current vanishes. You can then transform the Neumann condition of the previous PDE to

$$\mathbf{n} \cdot \left( \mu_0^{-1} \nabla A_z - \begin{bmatrix} -M_y \\ M_x \end{bmatrix} \right) = -\mathbf{n} \times (\mu_0^{-1} \nabla \times \mathbf{A} - \mathbf{M}) = -\mathbf{n} \times \mathbf{H} = \mathbf{0}$$

### MAGNETIC FIELD

The magnetic-field boundary condition

$$\mathbf{n} \times \mathbf{H} = \mathbf{n} \times \mathbf{H}_0$$

specifies the tangential component of the magnetic field at the boundary.

### SURFACE CURRENT

The surface current boundary condition

$$-\mathbf{n} \times \mathbf{H} = J_{sz} \mathbf{e}_z \quad \mathbf{n} \times (\mathbf{H}_1 - \mathbf{H}_2) = J_{sz} \mathbf{e}_z$$

specifies a surface current flowing in the  $z$  direction.

### ELECTRIC INSULATION

The electric-insulation boundary condition

$$\mathbf{n} \times \mathbf{H} = \mathbf{0}$$

sets the magnetic field to zero. The term *electric insulation* arises from the fact that this boundary condition makes the normal component of the electric current equal to zero.

### MAGNETIC POTENTIAL

The magnetic-potential boundary condition

$$A_z = A_{0z}$$

specifies the magnetic potential.

### MAGNETIC INSULATION

The magnetic-insulation boundary condition

$$A_z = 0$$

sets the magnetic potential to zero at the boundary. This boundary condition is also applicable at symmetry boundaries where the magnetic field is known to be tangential to the boundary.

The term *magnetic insulation* comes from the fact that this boundary condition makes the normal component of the magnetic field equal to zero. Thus the boundary is not truly insulating.

### CONTINUITY

The continuity boundary condition

$$\mathbf{n} \times (\mathbf{H}_1 - \mathbf{H}_2) = \mathbf{0}$$

is the natural boundary condition implying continuity of the tangential component of the magnetic field.

### *Point Sources*

---

You can specify that points in the geometry carry a current  $I_0$  flowing in the  $z$  direction.

### *Application Mode Variables*

---

For time-harmonic analysis, all fundamental field quantities are available for postprocessing and for use in equations and boundary conditions. The same holds true for the magnetic potential and the time average of energy and heating expressions.

At the edges of a modeled structure you have access to the magnetic potential, electric field, and tangential magnetic field for postprocessing. The energy flow in the normal direction is also available.

The expressions for some variables in the following table vary depending on the constitutive relationship as indicated in the Const. Rel. column; the abbreviations indicate the constitutive relationship as given in the second table.

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
Az	S		magnetic potential, $z$ component	$A_z$
mu0	S		permeability of a vacuum	$\mu_0$
mur	S	mur, Br	relative permeability	$\mu_r$



NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
mur	S	M	relative permeability	1
mur <sub>ij</sub>	S	mur, Br	relative permeability, $x_i x_j$ component	$\mu_{rij}$
mur <sub>ij</sub>	S	M	relative permeability, $x_i x_j$ component	1
mu	S		permeability	$\mu_0 \mu_r$
mu <sub>ij</sub>	S		permeability, $x_i x_j$ component	$\mu_0 \mu_{rij}$
sigma	S		electric conductivity	$\sigma$
deltaV	S		potential difference	$\Delta V$
L	S		length	$L$
M <sub>i</sub>	S	M	magnetization, $x_i$ component	$M_i$
M <sub>i</sub>	S	mur, Br	magnetization, $x_i$ component	$B_i / \mu_0 - H_i$
normM	S		magnetization, norm	$\sqrt{\mathbf{M} \cdot \mathbf{M}^*}$
Br <sub>i</sub>	S	mur	remanent flux density, $x_i$ component	0
Br <sub>i</sub>	S	M	remanent flux density, $x_i$ component	$\mu_0 M_i$
Br <sub>i</sub>	S	Br	remanent flux density, $x_i$ component	$B_{ri}$
normBr	S		remanent flux density, norm	$\sqrt{\mathbf{B}_r \cdot \mathbf{B}_r^*}$
J <sub>ez</sub>	S		external current density, $z$ component	$J_z^e$

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
$v_i$	S		velocity, $x_i$ component	$v_i$
normv	S		velocity, norm	$\sqrt{\mathbf{v} \cdot \mathbf{v}}$
Bx	S		magnetic flux density, $x$ component	$\frac{\partial A_z}{\partial y}$
By	S		magnetic flux density, $y$ component	$-\frac{\partial A_z}{\partial x}$
normB	S		magnetic flux density, norm	$\sqrt{\mathbf{B} \cdot \mathbf{B}^*}$
$H_i$	S	mur	magnetic field, $x_i$ component	$\mu_{rij}^{-1} B_j / \mu_0$
$H_i$	S	M	magnetic field, $x_i$ component	$B_i / \mu_0 - M_i$
$H_i$	S	Br	magnetic field, $x_i$ component	$\mu_{rij}^{-1} (B_j - B_{rj}) / \mu_0$
normH	S		magnetic field, norm	$\sqrt{\mathbf{H} \cdot \mathbf{H}^*}$
Jpz	S		potential current, $z$ component	$\sigma \Delta V / L$
Jvz	S		velocity current density, $z$ component	$\sigma (v_x B_y - v_y B_x)$
Jz	S		total current density, $z$ component	$J_z^e + J_z^v + J_z^p$
normJ	S		total current density, norm	$ J_z $
Wm	S		magnetic energy density	$\frac{\mathbf{H} \cdot \mathbf{B}}{2}$
Q	S		resistive heating	$J_z \left( v_x B_y - v_y B_x + \frac{\Delta V}{L} + \sigma^{-1} J_z^e \right)$
FLtzi	S		Lorentz force, $x_i$ component	$\mathbf{J} \times \mathbf{B}$

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
FsLtzi	B		Lorentz surface force, $x_i$ component	$\mathbf{J}_s \times \mathbf{B}$
Jsz	B		surface current density, $z$ component	$n_y H_x - n_x H_y$
unTi	B		Maxwell surface stress tensor, $x_i$ component, up side of boundary	$-\frac{1}{2}(\mathbf{H}_{up} \cdot \mathbf{B}_{up})n_{i down}$ $+ (\mathbf{n}_{down} \cdot \mathbf{H}_{up})B_{i up}$
dnTi	B		Maxwell surface stress tensor, $x_i$ component, down side of boundary	$-\frac{1}{2}(\mathbf{H}_{down} \cdot \mathbf{B}_{down})n_{i up}$ $+ (\mathbf{n}_{up} \cdot \mathbf{H}_{down})B_{i down}$

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
Aphidr	S		magnetic potential divided by $r$	$u$
Aphi	S		magnetic potential, $\varphi$ component	$ru$
Aphir	S		magnetic potential, $r$ derivative of $\varphi$ component	$r \frac{\partial u}{\partial r} + u$
Aphiz	S		magnetic potential, $z$ derivative of $\varphi$ component	$r \frac{\partial u}{\partial z}$
mu0	S		permeability of a vacuum	$\mu_0$
mur	S	mur, Br	relative permeability	$\mu_r$
mur	S	M	relative permeability	1

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
$\mu_{rij}$	S	mur, Br	relative permeability, $x_i x_j$ component	$\mu_{rij}$
$\mu_{rij}$	S	M	relative permeability, $x_i x_j$ component	1
$\mu$	S		permeability	$\mu_0 \mu_r$
$\mu_{ij}$	S		permeability, $x_i x_j$ component	$\mu_0 \mu_{rij}$
sigma	S		electric conductivity	$\sigma$
Vloop	S		loop potential	$V_{loop}$
$M_i$	S	M	magnetization, $x_i$ component	$M_i$
$M_i$	S	mur, Br	magnetization, $x_i$ component	$B_i / \mu_0 - H_i$
normM	S		magnetization, norm	$\sqrt{\mathbf{M} \cdot \mathbf{M}^*}$
$Br_i$	S	mur	remanent flux density, $x_i$ component	0
$Br_i$	S	M	remanent flux density, $x_i$ component	$\mu_0 M_i$
$Br_i$	S	Br	remanent flux density, $x_i$ component	$B_{ri}$
normBr	S		remanent flux density, norm	$\sqrt{\mathbf{B}_r \cdot \mathbf{B}_r^*}$
Jephi	S		external current density, $\varphi$ component	$J_\varphi^e$
$v_i$	S		velocity, $x_i$ component	$v_i$
normv	S		velocity, norm	$\sqrt{\mathbf{v} \cdot \mathbf{v}}$
normE	S		electric field, norm	$ E_\varphi $

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
Br	S		magnetic flux density, $r$ component	$-\frac{\partial A_\varphi}{\partial z}$
Bz	S		magnetic flux density, $z$ component	$u + \frac{\partial A_\varphi}{\partial r}$
normB	S		magnetic flux density, norm	$\sqrt{\mathbf{B} \cdot \mathbf{B}^*}$
Hi	S	mur	magnetic field, $x_i$ component	$\mu_{rij}^{-1} B_j / \mu_0$
Hi	S	M	magnetic field, $x_i$ component	$B_i / \mu_0 - H_i$
Hi	S	Br	magnetic field, $x_i$ component	$\mu_{rij}^{-1} (B_j - B_{rj}) / \mu_0$
normH	S		magnetic field, norm	$\sqrt{\mathbf{H} \cdot \mathbf{H}^*}$
Jpphi	S		loop current, $\varphi$ component	$\sigma V_{loop} / 2\pi r$
Jiphi	S		induced current density, $\varphi$ component	$\sigma E_\varphi$
Jvphi	S		velocity current density, $\varphi$ component	$\sigma (v_z B_r - v_r B_z)$
Jphi	S		total current density, $\varphi$ component	$J_\varphi^e + J_\varphi^v + J_\varphi^p$
normJ	S		total current density, norm	$ J_\varphi $
Wm	S		magnetic energy density	$\frac{\mathbf{H} \cdot \mathbf{B}}{2}$
Q	S		resistive heating	$J_\varphi \left( v_r B_z - v_z B_r + \frac{V_{loop}}{2\pi r} + \sigma^{-1} J_\varphi^e \right)$
Jsph	B		surface current density, $\varphi$ component	$n_r H_z - n_z H_r$

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
unT <sub>i</sub>	B		Maxwell surface stress tensor, $x_i$ component, up side of boundary	$-\frac{1}{2}(\mathbf{H}_{\text{up}} \cdot \mathbf{B}_{\text{up}})n_{i\text{down}}$ $+ (\mathbf{n}_{\text{down}} \cdot \mathbf{H}_{\text{up}})B_{i\text{up}}$
dnT <sub>i</sub>	B		Maxwell surface stress tensor, $x_i$ component, down side of boundary	$-\frac{1}{2}(\mathbf{H}_{\text{down}} \cdot \mathbf{B}_{\text{down}})n_{i\text{up}}$ $+ (\mathbf{n}_{\text{up}} \cdot \mathbf{H}_{\text{down}})B_{i\text{down}}$

**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: qa), for example, normH\_qa. (This does not apply to the dependent variables for the magnetic potential's  $z$  component.)

TABLE 5-2: ABBREVIATIONS FOR THE CONSTITUTIVE RELATIONSHIPS

ABBREVIATION	CONSTITUTIVE RELATIONSHIP
mur	$\mathbf{B} = \mu_0\mu_r\mathbf{H}$
M	$\mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M})$
Br	$\mathbf{B} = \mu_0\mu_r\mathbf{H} + \mathbf{B}_r$

# AC Power Electromagnetics

AC power electromagnetics problems are common when studying motors, transformers, and conductors carrying alternating currents.

You can use the *AC Power Electromagnetics* application mode for 2D in-plane and 2D axisymmetric models.

## *PDE Formulation*

---

To derive the equation system this mode solves, start with Ampère's law,

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t} = \sigma \mathbf{E} + \sigma \mathbf{v} \times \mathbf{B} + \mathbf{J}^e + \frac{\partial \mathbf{D}}{\partial t}.$$

Now assume time-harmonic fields and use the definitions of the potentials,

$$\begin{aligned}\mathbf{B} &= \nabla \times \mathbf{A} \\ \mathbf{E} &= -\nabla V - \frac{\partial \mathbf{A}}{\partial t}\end{aligned}$$

and combine them with the constitutive relationships  $\mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M})$  and  $\mathbf{D} = \epsilon_0 \mathbf{E} + \mathbf{P}$  to rewrite Ampère's law as

$$(j\omega\sigma - \omega^2\epsilon_0)\mathbf{A} + \nabla \times (\mu_0^{-1}\nabla \times \mathbf{A} - \mathbf{M}) - \sigma \mathbf{v} \times (\nabla \times \mathbf{A}) + (\sigma + j\omega\epsilon_0)\nabla V = \mathbf{J}^e + j\omega\mathbf{P}$$

In the 2D in-plane case there are no variations in the  $z$  direction, and the electric field is parallel to the  $z$ -axis. Therefore you can write  $\nabla V$  as  $-\Delta V/L$  where  $\Delta V$  is the potential difference over the distance  $L$ .

Now simplify these equations to

$$-\nabla \cdot \left( \mu_0^{-1} \nabla A_z - \begin{bmatrix} -M_y \\ M_x \end{bmatrix} \right) + \sigma \mathbf{v} \cdot \nabla A_z + (j\omega\sigma - \omega^2\epsilon_0)A_z = \sigma \frac{\Delta V}{L} + \mathcal{J}_z^e + j\omega P_z.$$

The axisymmetric case uses another form of the gradient of the electric potential,  $\nabla V = -V_{\text{loop}}/(2\pi r)$ , because the electric field is present only in the azimuthal direction. The above equation, in cylindrical coordinates, becomes

$$\begin{aligned}
& - \left( \left[ \frac{\partial}{\partial r} \quad \frac{\partial}{\partial z} \right] \cdot \left( r \mu_0^{-1} \begin{bmatrix} \frac{\partial u}{\partial r} \\ \frac{\partial u}{\partial z} \end{bmatrix} + \mu_0^{-1} \begin{bmatrix} 2 \\ 0 \end{bmatrix} u - \begin{bmatrix} M_z \\ -M_r \end{bmatrix} \right) \right) + r \sigma \left( \mathbf{v} \cdot \begin{bmatrix} \frac{\partial u}{\partial r} \\ \frac{\partial u}{\partial z} \end{bmatrix} \right) + r(\sigma j \omega - \omega^2 \varepsilon_0)u + 2\sigma v_r u \\
& = \sigma \frac{V_{\text{loop}}}{2\pi r} + J_\varphi^e + j\omega P_\varphi
\end{aligned}$$

The dependent variable  $u$  is the nonzero component of the magnetic potential divided by the radial coordinate  $r$ , so that

$$u = \frac{A_\varphi}{r}$$

The application mode performs this transformation to avoid singularities on the symmetry axis.

### *Application Scalar Variables*

The application scalar variables in this application mode are:

PROPERTY	NAME	DEFAULT	DESCRIPTION
$\mu_0$	mu0	$4\pi \cdot 10^{-7}$ H/m	Permeability of vacuum
$\varepsilon_0$	epsilon0	$8.854187817 \cdot 10^{-12}$ F/m	Permittivity of vacuum
$\nu$	nu	50 Hz	Frequency

The frequency occurs as a variable in time-harmonic problems.

### *Boundary and Interface Conditions*

The relevant interface condition is

$$\mathbf{n}_2 \times (\mathbf{H}_1 - \mathbf{H}_2) = \mathbf{J}_s$$

The natural boundary condition fulfills this equation if the surface current vanishes. You can transform the Neumann condition of this PDE into



$$\begin{aligned}
& \mathbf{n} \cdot \left[ \left( \mu_0^{-1} \nabla A_z - \begin{bmatrix} -M_y \\ M_x \end{bmatrix} \right)_1 - \left( \mu_0^{-1} \nabla A_z - \begin{bmatrix} -M_y \\ M_x \end{bmatrix} \right)_2 \right] \\
&= -\mathbf{n} \times [(\mu_0^{-1} \nabla \times \mathbf{A} - \mathbf{M})_1 - (\mu_0^{-1} \nabla \times \mathbf{A} - \mathbf{M})_2] \\
&= -\mathbf{n} \times (\mathbf{H}_1 - \mathbf{H}_2) = \mathbf{0}
\end{aligned}$$

### MAGNETIC FIELD

The magnetic-field boundary condition

$$\mathbf{n} \times \mathbf{H} = \mathbf{n} \times \mathbf{H}_0$$

specifies the tangential component of the magnetic field at the boundary.

### SURFACE CURRENT

The surface-current boundary condition

$$-\mathbf{n} \times \mathbf{H} = J_{sz} \mathbf{e}_z \quad \mathbf{n} \times (\mathbf{H}_1 - \mathbf{H}_2) = J_{sz} \mathbf{e}_z$$

lets you specify a surface current flowing in the  $z$  direction.

### ELECTRIC INSULATION

The electric-insulation boundary condition

$$\mathbf{n} \times \mathbf{H} = \mathbf{0}$$

sets the magnetic field to zero. The term *electric insulation* comes from the fact that this boundary condition makes the normal component of the electric current equal to zero.

### MAGNETIC POTENTIAL

The magnetic-potential boundary condition

$$A_z = A_{0z}$$

specifies the magnetic potential.

### MAGNETIC INSULATION

The magnetic insulation boundary condition

$$A_z = 0$$

sets the magnetic potential to zero at the boundary. This boundary condition is also applicable at symmetry boundaries where the magnetic field is known to be tangential to the boundary.

The term *magnetic insulation* comes from the fact that this boundary condition makes the normal component of the magnetic field zero. Thus the boundary is not truly insulating.

### CONTINUITY

The continuity boundary condition

$$\mathbf{n} \times (\mathbf{H}_1 - \mathbf{H}_2) = \mathbf{0}$$

is the natural boundary condition implying continuity of the tangential component of the magnetic field.

### *Point Sources*

---

You can specify that points in the geometry carry a current  $I_0$  flowing in the  $z$  direction.

### *Application Mode Variables*

---

For time-harmonic analysis, all the fundamental field quantities are available for postprocessing and for use in equations and boundary conditions. The same holds true for the magnetic potential and the time average of energy and heating expressions.

At the edges of the modeled structure you can also use the magnetic potential, electric field and tangential magnetic field for postprocessing. The energy flow in the normal direction is also available.

The expressions for some variables in the following table vary depending on the constitutive relationship as indicated in the Const. Rel. column; the abbreviations indicate the constitutive relationship as given in the second table.'

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
Az	S		magnetic potential, $z$ component	$A_z$
nu	S		frequency	$\nu$
omega	S		angular frequency	$2\pi\nu$

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
epsilon0	S		permittivity of a vacuum	$\epsilon_0$
mu0	S		permeability of a vacuum	$\mu_0$
mur	S	mur, Br	relative permeability	$\mu_r$
mur	S	M	relative permeability	1
mur <sub>ij</sub>	S	mur, Br	relative permeability, $x_i x_j$ component	$\mu_{rij}$
mur <sub>ij</sub>	S	M	relative permeability, $x_i x_j$ component	1
mu	S		permeability	$\mu_0 \mu_r$
mu <sub>ij</sub>	S		permeability, $x_i x_j$ component	$\mu_0 \mu_{rij}$
epsilon <sub>r</sub>	S	epsr, Dr	relative permittivity	$\epsilon_r$
epsilon <sub>r</sub>	S	P	relative permittivity	1
epsilon	S		permittivity	$\epsilon_0 \epsilon_r$
sigma	S		electric conductivity	$\sigma$
deltaV	S		potential difference	$\Delta V$
L	S		length	$L$
P <sub>z</sub>	S	P	electric polarization, $z$ component	$P_z e^{j\text{phase}}$
P <sub>z</sub>	S	epsr, Dr	electric polarization, $z$ component	$D_z - \epsilon_0 E_z$
Dr <sub>z</sub>	S	epsr	remanent displacement, $z$ component	0

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
Drz	S	P	remnant displacement, z component	$P_z$
Drz	S	Dr	remnant displacement, z component	$D_{rz}e^{j\text{phase}}$
Mi	S	M	magnetization, $x_i$ component	$M_i e^{j\text{phase}}$
Mi	S	mur, Br	magnetization, $x_i$ component	$B_i/\mu_0 - H_i$
normM	S		magnetization, norm	$\sqrt{\mathbf{M} \cdot \mathbf{M}^*}$
Br <sub>i</sub>	S	mur	remnant flux density, $x_i$ component	0
Br <sub>i</sub>	S	M	remnant flux density, $x_i$ component	$\mu_0 M_i$
Br <sub>i</sub>	S	Br	remnant flux density, $x_i$ component	$B_{ri} e^{j\text{phase}}$
normBr	S		remnant flux density, norm	$\sqrt{\mathbf{B}_r \cdot \mathbf{B}_r^*}$
Jez	S		external current density, z component	$J_z^e e^{j\text{phase}}$
vi	S		velocity, $x_i$ component	$v_i$
normv	S		velocity, norm	$\sqrt{\mathbf{v} \cdot \mathbf{v}}$
Ez	S		electric field, z component	$-j\omega A_z$
normE	S		electric field, norm	$ E_z $
Dz	S	epsr	electric displacement, z component	$\epsilon_0 \epsilon_r E_z$

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
Dz	S	P	electric displacement, z component	$\epsilon_0 E_z + P_z$
Dz	S	Dr	electric displacement, z component	$\epsilon_0 \epsilon_r E_z + D_{rz}$
normD	S		electric displacement, norm	$ D_z $
Bx	S		magnetic flux density, x component	$\frac{\partial A_z}{\partial y}$
By	S		magnetic flux density, y component	$-\frac{\partial A_z}{\partial x}$
normB	S		magnetic flux density, norm	$\sqrt{\mathbf{B} \cdot \mathbf{B}^*}$
Hi	S	mur	magnetic field, $x_i$ component	$\mu_{rij}^{-1} B_j / \mu_0$
Hi	S	M	magnetic field, $x_i$ component	$B_i / \mu_0 - M_i$
Hi	S	Br	magnetic field, $x_i$ component	$\mu_{rij}^{-1} (B_j - B_{rj}) / \mu_0$
normH	S		magnetic field, norm	$\sqrt{\mathbf{H} \cdot \mathbf{H}^*}$
Jpz	S		potential current, z component	$\sigma \Delta V / L$
Jiz	S		induced current density, z component	$\sigma E_z$
Jdz	S		displacement current density, z component	$j\omega D_z$
Jvz	S		velocity current density, z component	$\sigma (v_x B_y - v_y B_x)$

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
Jz	S		total current density, z component	$J_z^e + J_z^v + J_z^p + J_z^i + J_z^d$
normJ	S		total current density, norm	$ J_z $
Weav	S		time average electric energy density	$\frac{1}{4}\text{Re}(E_z D_z^*)$
Wmav	S		time average magnetic energy density	$\frac{1}{4}\text{Re}(\mathbf{H} \cdot \mathbf{B}^*)$
Wav	S		time average total energy density	$W_e^{\text{av}} + W_m^{\text{av}}$
Qav	S		time average resistive heating	$\frac{1}{2}\text{Re}\left(J_z\left(E_z^* + v_x B_y^* - v_y B_x^* + \frac{\Delta V^*}{L} + \sigma^{-1} J_z^{e*}\right)\right)$
Poxav	S		time average power flow, x component	$-\frac{1}{2}\text{Re}(E_z H_y^*)$
Poyav	S		time average power flow, y component	$\frac{1}{2}\text{Re}(E_z H_x^*)$
normPoav	S		time average power flow, norm	$\sqrt{\mathbf{S}^{\text{av}} \cdot \mathbf{S}^{\text{av}*}}$
Qmav	S		time average magnetic hysteresis losses	$-\frac{1}{2}\text{Re}(j\omega \mathbf{H} \cdot \mathbf{B}^*)$
FLtzi	S		Lorentz force, $x_i$ component	$\mathbf{J} \times \mathbf{B}$
FLtzavi	S		cycle average Lorentz force, $x_i$ component	$\frac{1}{2}(\mathbf{J} \times \mathbf{B}^*)_i$
FsLtzi	B		Lorentz surface force, $x_i$ component	$\mathbf{J}_s \times \mathbf{B}$
FsLtzavi	B		cycle average Lorentz surface force, $x_i$ component	$\frac{1}{2}(\mathbf{J}_s \times \mathbf{B}^*)_i$

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
Jsz	B		surface current density, z component	$n_y H_x - n_x H_y$
nPoav	B		time average power outflow	$\mathbf{n} \cdot \mathbf{S}^{\text{av}}$
unTiav	B		Maxwell surface stress tensor, $x_i$ component, upside of boundary	$-\frac{1}{4} \text{Re}(\mathbf{H}_{\text{up}} \cdot \mathbf{B}_{\text{up}}^*) n_{i\text{down}}$ $+\frac{1}{2} \text{Re}((\mathbf{n}_{\text{down}} \cdot \mathbf{H}_{\text{up}}) B_{i\text{up}}^*)$
dnTiav	B		Maxwell surface stress tensor, $x_i$ component, downside of boundary	$-\frac{1}{4} \text{Re}(\mathbf{H}_{\text{down}} \cdot \mathbf{B}_{\text{down}}^*) n_{i\text{up}}$ $+\frac{1}{2} \text{Re}((\mathbf{n}_{\text{up}} \cdot \mathbf{H}_{\text{down}}) B_{i\text{down}}^*)$

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
Aphidr	S		magnetic potential divided by $r$	$u$
Aphi	S		magnetic potential, $\varphi$ component	$ru$
Aphir	S		magnetic potential, $r$ derivative of $\varphi$ component	$r \frac{\partial u}{\partial r} + u$
Aphiz	S		magnetic potential, $z$ derivative of $\varphi$ component	$r \frac{\partial u}{\partial z}$
nu	S		frequency	$\nu$
omega	S		angular frequency	$2\pi\nu$
epsilon0	S		permittivity of vacuum	$\epsilon_0$
mu0	S		permeability of vacuum	$\mu_0$

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
mur	S	mur, Br	relative permeability	$\mu_r$
mur	S	M	relative permeability	1
mur <sub>ij</sub>	S	mur, Br	relative permeability, $x_i x_j$ component	$\mu_{rij}$
mur <sub>ij</sub>	S	M	relative permeability, $x_i x_j$ component	1
mu	S		permeability	$\mu_0 \mu_r$
mu <sub>ij</sub>	S		permeability, $x_i x_j$ component	$\mu_0 \mu_{rij}$
epsilon <sub>r</sub>	S	epsr, Dr	relative permittivity	$\epsilon_r$
epsilon <sub>r</sub>	S	P	relative permittivity	1
epsilon	S		permittivity	$\epsilon_0 \epsilon_r$
sigma	S		electric conductivity	$\sigma$
Vloop	S		loop potential	$V_{loop}$
Pphi	S	P	electric polarization, $\varphi$ component	$P_\varphi e^{j\text{phase}}$
Pphi	S	epsr, Dr	electric polarization, $\varphi$ component	$D_\varphi - \epsilon_0 E_\varphi$
Drphi	S	epsr	remanent displacement, $\varphi$ component	0
Drphi	S	P	remanent displacement, $\varphi$ component	$P_\varphi$
Drphi	S	Dr	remanent displacement, $\varphi$ component	$D_{r\varphi} e^{j\text{phase}}$



NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
$M_i$	S	M	magnetization, $x_i$ component	$M_i e^{j\text{phase}}$
$M_i$	S	mur, Br	magnetization, $x_i$ component	$B_i / \mu_0 - H_i$
normM	S		magnetization, norm	$\sqrt{\mathbf{M} \cdot \mathbf{M}^*}$
$Br_i$	S	mur	remanent flux density, $x_i$ component	0
$Br_i$	S	M	remanent flux density, $x_i$ component	$\mu_0 M_i$
$Br_i$	S	Br	remanent flux density, $x_i$ component	$Br_i e^{j\text{phase}}$
normBr	S		remanent flux density, norm	$\sqrt{\mathbf{B}_r \cdot \mathbf{B}_r^*}$
Jephi	S		external current density, $\varphi$ component	$J_\varphi^e e^{j\text{phase}}$
$v_i$	S		velocity, $x_i$ component	$v_i$
normv	S		velocity, norm	$\sqrt{\mathbf{v} \cdot \mathbf{v}}$
Ephi	S		electric field, $\varphi$ component	$-j\omega A_\varphi$
normE	S		electric field, norm	$ E_\varphi $
Dphi	S	epsr	electric displacement, $\varphi$ component	$\epsilon_0 \epsilon_r E_\varphi$
Dphi	S	P	electric displacement, $\varphi$ component	$\epsilon_0 E_\varphi + P_\varphi$
Dphi	S	Dr	electric displacement, $\varphi$ component	$\epsilon_0 \epsilon_r E_\varphi + D_{r\varphi}$

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
normD	S		electric displacement, norm	$ D_\phi $
Br	S		magnetic flux density, $r$ component	$-\frac{\partial A_\phi}{\partial z}$
Bz	S		magnetic flux density, $z$ component	$u + \frac{\partial A_\phi}{\partial r}$
normB	S		magnetic flux density, norm	$\sqrt{\mathbf{B} \cdot \mathbf{B}^*}$
Hi	S	mur	magnetic field, $x_i$ component	$\mu_{rij}^{-1} B_j / \mu_0$
Hi	S	M	magnetic field, $x_i$ component	$B_i / \mu_0 - M_i$
Hi	S	Br	magnetic field, $x_i$ component	$\mu_{rij}^{-1} (B_j - B_{rj}) / \mu_0$
normH	S		magnetic field, norm	$\sqrt{\mathbf{H} \cdot \mathbf{H}^*}$
Jpphi	S		loop current, $\phi$ component	$\sigma V_{\text{loop}} / 2\pi r$
Jiphi	S		induced current density, $\phi$ component	$\sigma E_\phi$
Jdphi	S		displacement current density, $\phi$ component	$j\omega D_\phi$
Jvphi	S		velocity current density, $\phi$ component	$\sigma(v_z B_r - v_r B_z)$
Jphi	S		total current density, $\phi$ component	$J_\phi^e + J_\phi^v + J_\phi^p + J_\phi^i + J_\phi^d$
normJ	S		total current density, norm	$ J_\phi $
Weav	S		time-average electric energy density	$\frac{1}{4} \text{Re}(E_\phi D_\phi^*)$

NAME	TYPE	CONST. REL.	DESCRIPTION	EXPRESSION
Wmav	S		time-average magnetic energy density	$\frac{1}{4}\text{Re}(\mathbf{H} \cdot \mathbf{B}^*)$
Wav	S		time-average total energy density	$W_e^{\text{av}} + W_m^{\text{av}}$
Qav	S		time-average resistive heating	$\frac{1}{2}\text{Re}\left(J_\phi\left(E_\phi^* + v_r B_z^* - v_z B_r^* + \frac{V_{\text{loop}}^*}{2\pi r} + \sigma^{-1} J_\phi^e\right)\right)$
Porav	S		time-average power flow, $r$ component	$-\frac{1}{2}\text{Re}(E_\phi H_z^*)$
Pozav	S		time-average power flow, $z$ component	$\frac{1}{2}\text{Re}(E_\phi H_z^*)$
normPoav	S		time-average power flow, norm	$\sqrt{\mathbf{S}^{\text{av}} \cdot \mathbf{S}^{\text{av}*}}$
Jsph	B		surface current density, $\phi$ component	$n_r H_z - n_z H_r$
nPoav	B		time-average power outflow	$\mathbf{n} \cdot \mathbf{S}^{\text{av}}$
unTiav	B		Maxwell surface stress tensor, $x_i$ component, upside of boundary	$-\frac{1}{4}\text{Re}(\mathbf{H}_{\text{up}} \cdot \mathbf{B}_{\text{up}}^*)n_{i\text{down}} + \frac{1}{2}\text{Re}((\mathbf{n}_{\text{down}} \cdot \mathbf{H}_{\text{up}})B_{i\text{up}}^*)$
dnTiav	B		Maxwell surface stress tensor, $x_i$ component, downside of boundary	$-\frac{1}{4}\text{Re}(\mathbf{H}_{\text{down}} \cdot \mathbf{B}_{\text{down}}^*)n_{i\text{up}} + \frac{1}{2}\text{Re}((\mathbf{n}_{\text{up}} \cdot \mathbf{H}_{\text{down}})B_{i\text{down}}^*)$

---

**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: qa), for example, Jz\_qa. (This does not apply to the dependent variable for the magnetic potential.)

---

TABLE 5-3: ABBREVIATIONS FOR THE CONSTITUTIVE RELATIONSHIPS

ABBREVIATION	CONSTITUTIVE RELATIONSHIP
mur	$\mathbf{B} = \mu_0 \mu_r \mathbf{H}$
M	$\mathbf{B} = \mu_0 (\mathbf{H} + \mathbf{M})$
Br	$\mathbf{B} = \mu_0 \mu_r \mathbf{H} + \mathbf{B}_r$

# Example—Electric Sensor

This is a model from electric impedance tomography, a method of imaging the interior permittivity distribution of an object by measuring current and voltage at the surface. The technique is used in, for example, medical diagnosis. Because different organs have different properties, you can “see” the organs and their movement from the outside.

The model shows how you can determine the shape and the placement of small objects with different material properties inside a closed box from outside. Applying a potential difference on the boundaries of the box gives rise to a surface charge density that varies depending on the permittivity distribution inside the box. By looking at the surface charge density you can therefore see the shape of the different materials inside the box.

## *Model Definition*

---

This model solves Gauss’ law with  $\rho = 0$ .

$$-\nabla \cdot (\epsilon_0 \epsilon_r \nabla V) = \rho$$

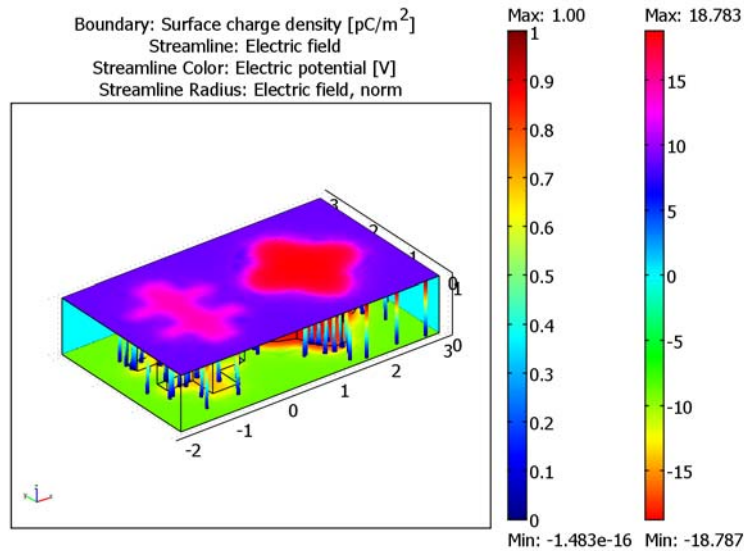
The box contains air with  $\epsilon_r$  equal to 1 and the different objects are made of material with different values of the relative permittivity,  $\epsilon_r$ : 1, 2, and 3.

To get a voltage difference, set  $V = 0$  on the bottom and  $V = 1$  on top of the box. On all the other boundaries, use an electric insulation condition:  $\mathbf{n} \cdot \mathbf{D} = 0$ .

## *Results and Discussion*

---

The surface charge density is higher above material with higher permittivity as expected. You can clearly see the shape of the figures inside the box on the top surface in the following plot.



Inside the geometry the streamlines show how the electric field varies. The gradient of the electric field is lower in media with larger value of the permittivity.

---

**Model Library path:** COMSOL\_Multiphysics/Electromagnetics/  
 electric\_sensor

---

### *Modeling Using the Graphical User Interface*

---

- 1 Open COMSOL Multiphysics.
- 2 In the **Model Navigator**, change the space dimension to **3D** and select **COMSOL Multiphysics>Electromagnetics>Electrostatics** in the list of COMSOL Multiphysics application modes.
- 3 Click **OK**.

#### **GEOMETRY MODELING**

- 1 Select **Work-Plane Settings** from the **Draw** menu. Set **z** to 0.1 and click **OK**.

- 2 From the **Options** menu, choose **Axes/Grid Settings**.
- 3 Enter the following data; when done, click **OK** (to enter the spacings on the Grid page, first clear the **Auto** check box):

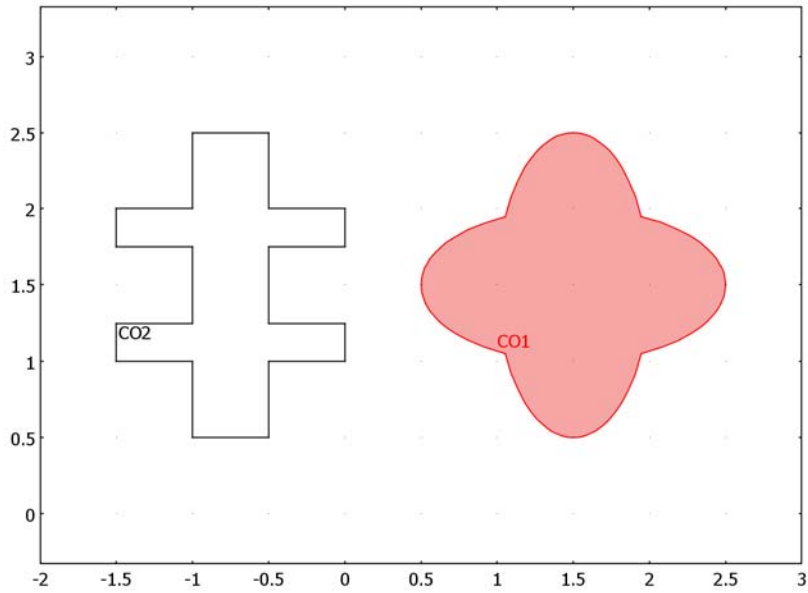
AXIS PAGE	
x min	-2
x max	3
y min	0
y max	3
GRID PAGE	
x spacing	0.5
y spacing	0.5

- 4 Draw a rectangle by clicking on the **Rectangle/Square** button and then clicking at the points  $(-1, 0.5)$  and  $(-0.5, 2.5)$ .
- 5 Specify two rectangles by choosing **Specify Objects>Rectangle** from the **Draw** menu. Use the following data:

	R2	R3
Width	1.5	1.5
Height	0.25	0.25
x	-1.5	-1.5
y	1	1.75

- 6 Select all objects by pressing Ctrl+A. On the Draw toolbar, click the **Union** button and then the **Delete Interior Boundaries** button.
- 7 Draw an ellipse by pressing the **Ellipse/Circle (Centered)** button. Click  $(1.5, 1.5)$  and  $(0.5, 1)$ .
- 8 Draw another ellipse clicking at the points  $(1.5, 1.5)$  and  $(1, 0.5)$ .
- 9 From the **Draw** menu or the Draw toolbar, open the **Create Composite Object** dialog box and select **E1** and **E2**. Clear the **Keep interior boundaries** check box, and click **OK**.

- 10 Select **Extrude** from the **Draw** menu, select both objects and enter 0.8 in the **Distance** edit field. Click **OK**.



- 11 Click the **Block** button. Enter the following dimensions and base point coordinates to create a block:

<b>BLOCK</b>	<b>BLK1</b>	
	Length	Axis base point
x	5	-2
y	3	0
z	1	0

- 12 Click the **Zoom Extents** button on the Main toolbar.

### PHYSICS SETTINGS

#### *Boundary Conditions*

- 1 Open the **Boundary Settings** dialog box from the **Physics** menu.
- 2 Press **Ctrl+A** to select all boundaries. Set the boundary condition to **Zero charge/Symmetry**.
- 3 Select Boundary 3 and set the boundary condition to **Ground**.



- 4 Select Boundary 4 and set the boundary condition to **Electric potential** and  $V_0$  to 1.
- 5 Click **OK**.

#### *Subdomain Settings*

In the **Subdomain Settings** dialog box, set the following values of  $\epsilon_r$ :

SETTING	SUBDOMAIN 1	SUBDOMAIN 2	SUBDOMAIN 3
$\epsilon_r$	1	2	3

#### **MESH GENERATION**

- 1 Open the **Free Mesh Parameters** dialog box and select **Fine** from the **Predefined mesh sizes** list on the **General** tab. Click **OK** to close the dialog box.
- 2 Click the **Initialize Mesh** button on the Main toolbar to initialize the mesh.

#### **COMPUTING THE SOLUTION**

Solve the model by clicking the **Solve** button on the Main toolbar.

#### **POSTPROCESSING AND VISUALIZATION**

- 1 Open the **Plot Parameters** dialog box.
- 2 On the **General** page clear the **Slice** check box and select the **Boundary** and **Streamline** check boxes.
- 3 Click the **Boundary** tab. In the **Boundary data** area, select **Surface charge density** from the **Predefined quantities** list. In the **Unit** edit field, type  $\text{pC}/\text{m}^2$ .
- 4 In the **Boundary color** area, select **hsv** from the **Colormap** list.
- 5 On the **Streamline** page, click the **Line Color** tab.
- 6 Click the **Use expression** button. Then click the **Color Expression** button and set the color data to **Electric potential** in the **Predefined quantities** list. Click **OK**.
- 7 Click the **Start Points** tab. Type 100 in the **Number of start points** edit field.
- 8 Select **Tube** from the **Line type** list.
- 9 Click the **Tube Radius** button and set the **Radius data** to **Electric field, norm**. Click **OK**.
- 10 Click **OK**.

To see the streamlines inside the box you must suppress some of the boundaries:

- 1 On the **Options** menu, point to **Suppress** and then click **Suppress Boundaries**.
- 2 In the dialog box select Boundaries 1, 2, and 6–37. Click **OK**.
- 3 Open the **Plot Parameters** dialog box and click **OK**.

# Example—A Permanent Magnet

As an example of a magnetostatics problem, consider how to model a horseshoe-shaped permanent magnet. It consists of a ferromagnetic material, but the two end sections, often painted red and white, are premagnetized in opposite directions. The magnetic field pattern around such a permanent magnet is well known.

The domain consists of four regions:

- Three parts of the permanent magnet—the two premagnetized ends and the curved midsection
- The air surrounding the magnet

The permeability  $\mu$  in air equals  $\mu_0 = 4\pi \cdot 10^{-7}$  H/m. Because the magnet is made of a ferromagnetic material, its relative permeability normally depends on the strength of the magnetic field, but in this model it is a constant with a value of 5000. The premagnetization adds a magnetization vector, pointing in the positive  $x$  direction at the upper end and in the negative  $x$  direction in the lower end. The magnitude of the magnetization is 750 kA/m.

It is reasonable to neglect the field at the boundaries of the computational domain, thus leading to the Dirichlet boundary condition  $\mathbf{A} = \mathbf{0}$  on the exterior boundary.

---

**Model Library path:** COMSOL\_Multiphysics/Electromagnetics/  
permanent\_magnet

---

## *Modeling Using the Graphical User Interface*

---

### **MODEL NAVIGATOR**

- 1 Go to the Model Navigator and select **2D** in the **Space dimension** list.
- 2 Open the **COMSOL Multiphysics>Electromagnetics** folder and then select **Magnetostatics** in the list of application modes.
- 3 Use the default quadratic Lagrange elements.
- 4 Click **OK**.

## OPTIONS AND SETTINGS

- 1 Enter the following values in the **Axes/Grid Settings** dialog box. To set the grid spacing, click the **Grid** tab and clear the **Auto** check box.

AXIS		GRID	
x min	-3	x spacing	0.2
x max	3	Extra x	
y min	-2	y spacing	0.2
y max	2	Extra y	

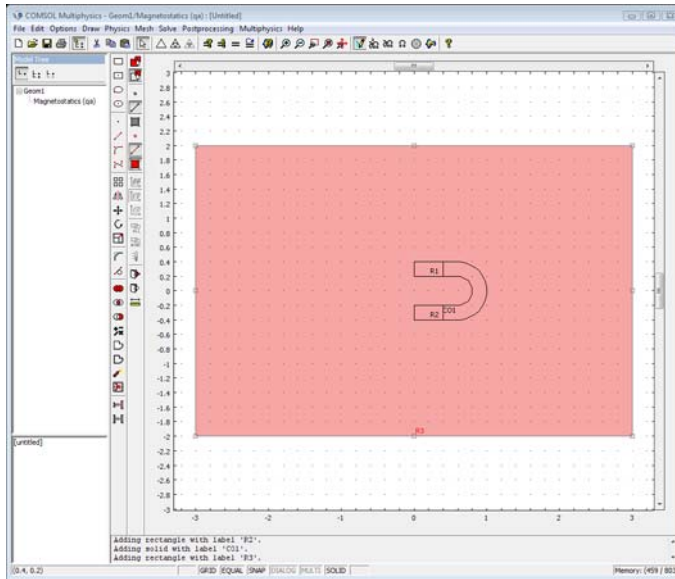
- 2 Enter these names and expressions in the **Constants** dialog box:

NAME	EXPRESSION
murFe	5000
Mpre	750000

## GEOMETRY MODELING

Start by modeling the upper and the lower premagnetized parts of the magnet.

- 1 Click the **Rectangle/Square** button and draw a rectangle from  $(0, 0.2)$  to  $(0.4, 0.4)$ .
- 2 Click the **Rectangle/Square** button and draw a rectangle from  $(0, -0.4)$  to  $(0.4, -0.2)$ .  
Next draw the rest of the magnet using boundary modeling:
- 3 Click the **Line** button, and click the points  $(0.4, 0.2)$ ,  $(0.4, 0.4)$ , and  $(0.6, 0.4)$ .
- 4 Click the **2nd Degree Bézier Curve** button, then click at the points  $(1, 0.4)$ ,  $(1, 0)$ ,  $(1, -0.4)$ , and  $(0.6, -0.4)$ .
- 5 Click **Line** and then click at the points  $(0.4, -0.4)$ ,  $(0.4, -0.2)$ , and  $(0.6, -0.2)$ .
- 6 Click **2nd Degree Bézier Curve** and then click at the points  $(0.8, -0.2)$ ,  $(0.8, 0)$ ,  $(0.8, 0.2)$ , and  $(0.6, 0.2)$ .
- 7 Close the region with the right mouse button.  
The last step in creating the geometry is to model the surrounding air.
- 8 Draw a rectangle from  $(-3, -2)$  to  $(3, 2)$ .



## PHYSICS SETTINGS

### Boundary Conditions

Use the default magnetic insulation boundary condition everywhere.

### Subdomain Settings

The relative permeability for the magnet is 5000, and the magnetization vector is  $(M_{\text{pre}}, 0)$  in the upper premagnetized part and  $(-M_{\text{pre}}, 0)$  in the lower part.

- 1 Go to the **Physics** menu and choose **Subdomain Settings**.
- 2 Enter the PDE coefficients as shown in the following table. For Subdomains 1 and 4 use the default constitutive relationship  $\mathbf{B} = \mu_0\mu_r\mathbf{H}$ , and for Subdomains 2 and 3 use the constitutive relationship  $\mathbf{B} = \mu_0\mathbf{H} + \mu_0\mathbf{M}$ .

SETTINGS	SUBDOMAIN 1	SUBDOMAIN 2	SUBDOMAIN 3	SUBDOMAIN 4
$\mu_r$	1			murFe
$M_x$		-Mpre	Mpre	
$M_y$		0	0	

## MESH GENERATION

To resolve the field at the ends of the magnets, use a smaller mesh size at those boundaries:

- 1 From the **Mesh** menu, choose **Free Mesh Parameters**.
- 2 Click the **Boundary** tab.
- 3 Select Boundaries 4 and 7 (the ends of the magnet).
- 4 Type 0.01 in the **Maximum element size** edit field.
- 5 Click **OK**.
- 6 Click the **Initialize Mesh** button on the Main toolbar to create the mesh.

## COMPUTING THE SOLUTION

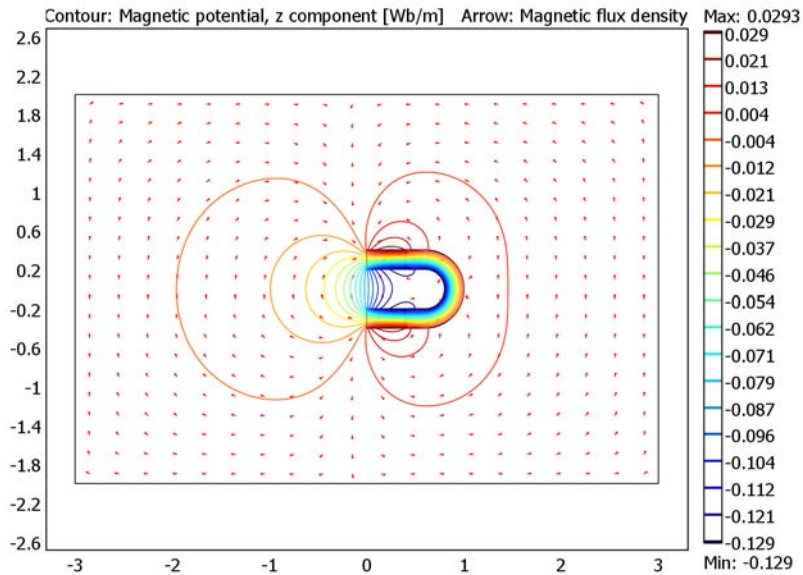
Use the adaptive solver to improve the solution accuracy.

- 1 Go to the **Solve** menu and choose **Solver Parameters**.
- 2 In the **Solver Parameters** dialog box, select the **Adaptive mesh refinement** check box underneath the **Solver** list.
- 3 Click **OK**.
- 4 Click the **Solve** button.

## POSTPROCESSING AND VISUALIZATION

Use the streamline plot to visualize the magnetic field using field lines where the distance between the lines is inversely proportional to the magnetic field strength.

- 1 Open the **Plot Parameters** dialog box.
- 2 Select the **Contour** and **Arrow** check boxes and clear the **Surface** check box in the **Plot type** area on the **General** page.
- 3 Click the **Contour** tab.
- 4 Select **Magnetic potential, z component** from the **Predefined quantities** list.
- 5 Click the **Arrow** tab.
- 6 Select **Magnetic flux density** from the **Predefined quantities** list.
- 7 Type 20 as **Number of points** in the **x points** and **y points** edit fields in the **Arrow positioning** area.
- 8 Select **Normalized** from the **Arrow length** list.
- 9 Clear the **Auto** check box and type **0.2** in the **Scale factor** edit field.
- 10 Click **OK**.



The resulting plot shows the well-known field pattern.

### *Alternative Modeling Approach*

An alternate way to model a permanent magnet is to set surface currents on the premagnetized parts. To do so, change the model from the previous discussion in the following fashion:

#### **PHYSICS SETTINGS**

##### *Boundary Conditions*

- 1 Select the **Interior boundaries** check box in the **Boundary Settings** dialog box.
- 2 Enter these boundary coefficients:

SETTINGS	BOUNDARIES 5, 9	BOUNDARIES 6, 8
Type	Surface current	Surface current
$J_{sz}$	Mpre	-Mpre

### *Subdomain Settings*

Modify the subdomain settings to remove the magnetization:

SETTING	SUBDOMAINS 2, 3
$M_x$	0

### **COMPUTING THE SOLUTION**

Go to the **Solver Parameters** dialog box and clear the **Adaptive mesh refinement** check box to turn off the adaptive solver. Simply use the mesh obtained by solving the previous problem. Solve the problem again.

### **POSTPROCESSING AND VISUALIZATION**

The solution plot reveals a result identical to the one where you modeled the premagnetization using a magnetization vector.

### *References*

---

1. D.K. Cheng, *Field and Wave Electromagnetics*, Addison-Wesley, Reading, MA, 1989.
2. J. Jin, *The Finite Element Method in Electromagnetics*, John Wiley & Sons, New York, 1993.
3. B.D. Popovic, *Introductory Engineering Electromagnetics*, Addison-Wesley, Reading, MA, 1971.





## Fluid Mechanics

This chapter explains how to use the Incompressible Navier-Stokes application mode for the modeling and simulation of fluid mechanics and fluid statics. Note that the engineering community often uses the term *CFD*, *computational fluid dynamics*, to refer to the numerical simulation of fluids. This chapter concludes with step-by-step instructions on how to model a common benchmark problem: flow over a backward step in the absence of external forces.

# The Navier-Stokes Application Mode

Fluid mechanics deals with studies of gases and liquids either in motion (*fluid dynamics*) or at rest (*fluid statics*). When studying liquid flows, it is often safe to assume that the material's density is constant or almost constant. You then have an *incompressible fluid flow*. Using the Incompressible Navier-Stokes application mode you can solve transient and steady-state models of incompressible fluid dynamics.

In the Model Navigator you find two entry points for **Incompressible Navier-Stokes**. These are **Steady-state analysis**, and **Transient analysis**.

---

**Note:** The optional Chemical Engineering Module contains more extensive application modes for incompressible Navier-Stokes problems including non-Newtonian flow and turbulence modeling using the  $k$ - $\epsilon$  and  $k$ - $\omega$  models. It also supplies application modes for non-isothermal and weakly compressible flow, swirl flow, multiphase flow, and the Brinkman equations and Darcy's law for flow in porous media.

---

## *Variables and Space Dimension*

---

The Incompressible Navier-Stokes application mode solves for the pressure  $p$  and the velocity vector components. It is available for 2D, 2D axisymmetric, and 3D geometries.

## *PDE Formulation and Equations*

---

COMSOL Multiphysics uses a generalized version of the Navier-Stokes equations to allow for variable viscosity.

Starting with the momentum balance in terms of stresses, the generalized equations in terms of transport properties and velocity gradients are

$$\rho \frac{\partial \mathbf{u}}{\partial t} - \nabla \cdot [\eta(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)] + \rho(\mathbf{u} \cdot \nabla) \mathbf{u} + \nabla p = \mathbf{F} \quad (6-1)$$
$$\nabla \cdot \mathbf{u} = 0$$

The first equation is the *momentum transport equations*, and the second is the *equation of continuity* for incompressible fluids. The following variables and parameters appear in the equations:

- $\eta$  is the dynamic viscosity
- $\rho$  is the density
- $\mathbf{u}$  is the velocity field
- $p$  is the pressure
- $\mathbf{F}$  is a volume force field such as gravity

These application modes are general enough to account for all types of incompressible flow. In practice, though, successful analysis of turbulent flows requires simplifications of the description of transport of momentum.

### *Subdomain Settings*

---

The subdomain quantities are:

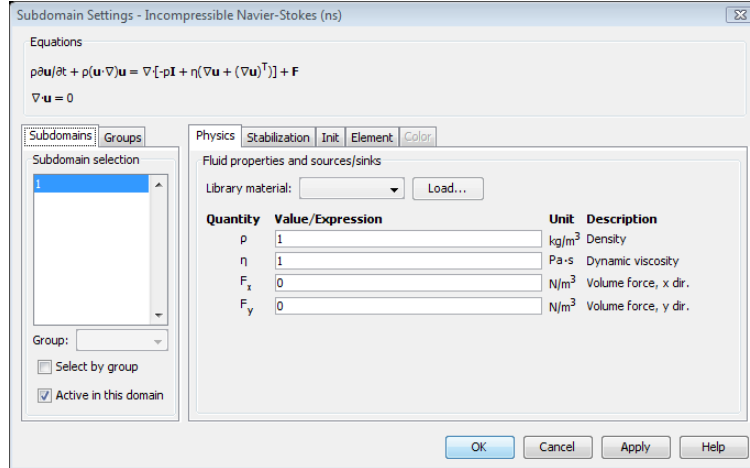
PARAMETER	VARIABLE	DESCRIPTION
$\rho$	rho	Density
$\eta$	eta	Dynamic viscosity
$\mathbf{F}$	F	Volume force

**Density** This material property specifies the fluid density.

**Dynamic Viscosity** This term describes the relationship between the shear stresses in a fluid and the shear rate. Intuitively, water and air have a low viscosity, and substances often described as thick, such as oil, have a higher viscosity. You can describe some non-Newtonian fluid by defining a shear-rate dependent viscosity. Examples of non-Newtonian fluids include yoghurt, paper pulp, and polymer suspensions.

**Volume Force** The volume force vector,  $\mathbf{F} = (F_x, F_y, F_z)$ , describes a distributed force field such as gravity. The unit of the volume force is force/volume.

You specify the subdomain properties in the **Subdomain Settings** dialog box.



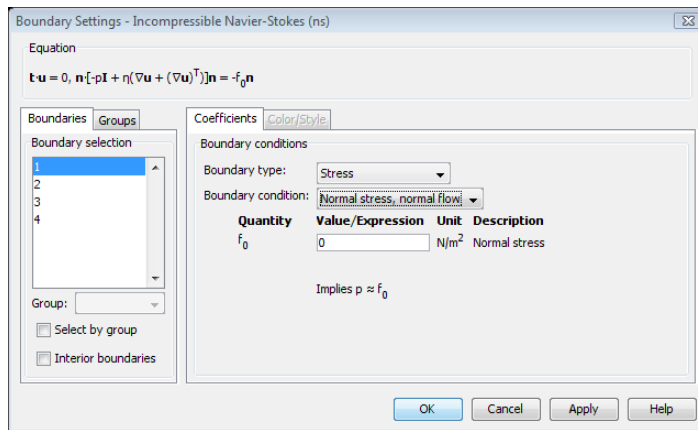
## Boundary Conditions

The boundary conditions for the Incompressible Navier-Stokes application mode are grouped into the following types:

- Wall
  - No slip (default)
  - Slip
  - Sliding wall
  - Moving/leaking wall
- Inlet
  - Velocity (default)
  - Pressure, no viscous stress
- Outlet
  - Velocity
  - Pressure
  - Pressure, no viscous stress (default)
  - No viscous stress
  - Normal stress

- Symmetry boundary
  - Symmetry (default)
  - Axial symmetry (2D axisymmetry only)
- Open boundary
  - Normal stress (default)
  - No viscous stress
- Stress
  - General stress (default)
  - Normal stress
  - Normal stress, normal flow

You specify a boundary condition in the **Boundary Settings** dialog box, where you first select the appropriate **Boundary type** and then a **Boundary condition**.



*Figure 6-1: Boundary Settings dialog box for the Incompressible Navier-Stokes application mode.*

If a mathematical formulation describes more than one type of physical boundary condition, it can appear in more than one boundary type. However, every possible use of a single mathematical formulation cannot be covered. Hence, the boundary types should be regarded as guidelines, not as restrictions on the applicability of the formulations.

The theory of most boundary conditions can be found in Ref. 1.

## WALL

These boundary conditions describe the existence of a solid wall.

### *No Slip*

This is the standard and default boundary condition for a stationary solid wall. The condition prescribes

$$\mathbf{u} = \mathbf{0}$$

that is, that the fluid at the wall is not moving.

### *Moving/Perforated Wall*

If the wall moves, so must the fluid. Hence, this boundary condition prescribes

$$\mathbf{u} = \mathbf{u}_w$$

Note that setting this boundary condition does not automatically cause the associated wall to move. The section “The Moving Mesh Application Mode” on page 463 of the *COMSOL Multiphysics Modeling Guide* describes how to set up a model with moving boundaries.

You can also use the Moving/perforated wall boundary condition to simulate a wall where fluid is leaking into or leaving through a perforated wall.

### *Sliding Wall*

If you use this boundary condition, the wall is assumed to behave like a conveyor belt, that is, that the surface is sliding in its tangential direction. The wall does not have to actually move in the coordinate system.

In two space dimensions (2D), the tangential direction is unambiguously defined by the direction of the boundary. However, the situation becomes more complicated in 3D. For this reason, this boundary condition has slightly different definitions in the different space dimensions.

**2D and Axial Symmetry** The velocity is given as a scalar  $U_w$  and the condition prescribes

$$\begin{aligned}\mathbf{u} \cdot \mathbf{n} &= 0 \\ \mathbf{u} \cdot \mathbf{t} &= U_w\end{aligned}$$

where  $\mathbf{t} = (-n_y, n_x)$  for 2D and  $\mathbf{t} = (-n_z, n_r)$  for axial symmetry.

**3D** The velocity is set equal to a given vector  $\mathbf{u}_w$  projected onto the boundary plane:

$$\mathbf{u} = \mathbf{u}_w - (\mathbf{n} \cdot \mathbf{u}_w)\mathbf{n}$$

### *Slip*

The slip condition assumes that there are no viscous effects at the slip wall and hence, no boundary layer develops. From a modeling point of view, this may be a reasonable approximation if the important effect of the wall is to prevent fluid from leaving the domain. Mathematically, the constraint can be formulated as:

$$\mathbf{u} \cdot \mathbf{n} = 0, \quad \mathbf{t} \cdot (-p\mathbf{I} + \eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T))\mathbf{n} = 0$$

where  $\mathbf{t}$  is a tangential vector to the boundary.

### **INLET**

This boundary type contains different ways to specify conditions on a boundary where the fluid is supposed to enter the domain. Notice that the formulations contained in this boundary type all appear, some of them slightly modified, in the Outflow boundary type as well. Hence, there is nothing in the mathematical formulations that prevents a fluid from leaving the domain through boundaries where you have specified the Inlet boundary type.

### *Velocity*

This boundary condition offers two ways to specify an inlet velocity. The first is to set the velocity equal to a given vector  $\mathbf{u}_0$ :

$$\mathbf{u} = \mathbf{u}_0$$

The other is to specify a normal inflow velocity:

$$\mathbf{u} = -\mathbf{n}U_0$$

Note that the boundary normal,  $\mathbf{n}$ , points out of the domain.

### *Pressure, No Viscous Stress*

This boundary condition specifies vanishing viscous stress along with a Dirichlet condition on the pressure:

$$\eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\mathbf{n} = \mathbf{0}, \quad p = p_0$$

It is a numerically stable boundary condition that admits total control of the pressure level along the entire boundary. However, if the inflow is not normal to the boundary, this condition is an overspecification. In the case that your solution turns out to have a non-normal inflow velocity, there are two choices. Either, move the boundary farther

away to a location where the inflow is normal to the boundary or, use a stress type boundary condition described on page 143.

Note that this condition is identical to the Pressure, no viscous stress condition for Outflow boundaries. Hence, depending on the pressure field in the rest of the subdomain, a boundary with this condition can very well become an outflow boundary.

### **OUTFLOW**

This boundary type contains different ways to specify conditions on a boundary where the fluid exits the domain. Note that all of the formulations in this type can be found, possibly slightly modified, in other boundary types as well. Hence, there is nothing in the mathematical formulations that prevent a fluid from entering the domain through boundaries where you have set the Outflow boundary type.

Setting outlet conditions for the Navier-Stokes equations is not a trivial task. A general rule of thumb, however, is that if there is something interesting happening at an outflow boundary, extend the computational domain to include this phenomenon.

#### *Velocity*

This boundary condition offers two ways to specify an outlet velocity. The first is to set the velocity equal to a given vector  $\mathbf{u}_0$ :

$$\mathbf{u} = \mathbf{u}_0$$

The other is to specify a normal outlet velocity:

$$\mathbf{u} = \mathbf{n}U_0$$

Observe that the boundary normal,  $\mathbf{n}$ , is pointing out of the domain.

#### *Pressure, No Viscous Stress*

This boundary condition specifies vanishing viscous stress along with a Dirichlet condition on the pressure:

$$\eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\mathbf{n} = \mathbf{0}, \quad p = p_0$$

It is a numerically stable boundary condition that admits total control of the pressure level at the whole boundary. However, if the outflow is not normal to the boundary, this condition is an overspecification. In the case that your solution turns out to have a non-normal outflow velocity, there are two choices. Either move the boundary



farther away to a location where the outflow is normal to the boundary or use a stress type boundary condition described on page 143.

Note that this condition is identical to the Pressure, no viscous stress condition for Inflow boundaries. Hence, depending on the pressure field in the rest of the subdomain, a boundary with this condition can very well become an inflow boundary.

#### *Pressure*

This boundary condition prescribes only a Dirichlet condition for the pressure:

$$p = p_0$$

Use this boundary condition only for high Reynolds number outflow boundaries; that is, boundaries where the cell Reynolds number  $\text{Re}^c = \rho|\mathbf{u}|h/(2\eta) \gg 1$  ( $h$  is the local mesh element size). It is far less stable than the Pressure, no viscous stress boundary condition, but it is consistent with a non-normal outflow velocity. The cell Reynolds number is available for postprocessing using the default name `cellRe_ns`.

#### *No Viscous Stress*

Prescribes vanishing viscous stress:

$$\eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\mathbf{n} = \mathbf{0}$$

This condition can be useful in some situations because it does not impose any constraint on the pressure. A typical example is a model with volume forces that give rise to pressure gradients that are hard to prescribe in advance. It should however be combined with a point constraint on the pressure to be numerically stable (see “Point Settings” on page 144).

#### *Normal Stress*

The total stress on the boundary is set equal to a stress vector of magnitude,  $f_0$ , oriented in the negative normal direction:

$$(-p\mathbf{I} + \eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T))\mathbf{n} = -f_0\mathbf{n}$$

This implies that the total stress in the tangential direction is zero. This boundary condition implicitly sets a constraint on the pressure that for 2D flows can be written

$$p = 2\eta\frac{\partial u_n}{\partial n} + f_0 \tag{6-2}$$

If  $\partial u_n/\partial n$  is small, Equation 6-2 can be interpreted as  $p \approx f_0$ .

### SYMMETRY BOUNDARY

Prescribes no penetration and vanishing shear stresses:

$$\mathbf{u} \cdot \mathbf{n} = 0, \quad \mathbf{t} \cdot (-p\mathbf{I} + \eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T))\mathbf{n} = 0$$

In 2D Axial Symmetry, the above formulation is called **Symmetry**.

#### *Axial Symmetry*

This boundary condition is only available in 2D Axial Symmetry. Use it on all boundaries with coordinate  $r = 0$ . It prescribes  $u_r = 0$  and vanishing stresses in the  $z$  direction.

### OPEN BOUNDARY

You can use this boundary type on boundaries that are open to large volumes of fluid. Fluid can both enter and leave the domain on boundaries with this type of condition.

#### *No Viscous Stress*

Prescribes vanishing viscous stress:

$$\eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)\mathbf{n} = \mathbf{0}$$

This condition can be useful in some situations because it does not impose any constraint on the pressure. A typical example is a model with volume forces that give rise to pressure gradients that are hard to prescribe in advance. It should however be combined with a point constraint on the pressure to be numerically stable (see “Point Settings” on page 144).

#### *Normal Stress*

The total stress on the boundary is set equal to a stress vector of magnitude,  $f_0$ , oriented in the negative normal direction:

$$(-p\mathbf{I} + \eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T))\mathbf{n} = -f_0\mathbf{n}$$

This implies that the total stress in the tangential direction is zero. This boundary condition implicitly sets a constraint on the pressure that for 2D flows can be written

$$p = 2\eta \frac{\partial u_n}{\partial n} + f_0 \quad (6-3)$$

If  $\partial u_n / \partial n$  is small, Equation 6-3 can be interpreted as  $p \approx f_0$ .

## STRESS

This type of boundary condition represents a very general class of conditions also known as traction boundary conditions.

### General Stress

The total stress on the boundary is set equal to a given stress  $\mathbf{F}$ :

$$(-p\mathbf{I} + \eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T))\mathbf{n} = \mathbf{F}$$

This boundary condition implicitly sets a constraint on the pressure that for 2D flows can be written

$$p = 2\eta\frac{\partial u_n}{\partial n} - \mathbf{n} \cdot \mathbf{F} \quad (6-4)$$

If  $\partial u_n/\partial n$  is small, Equation 6-4 can be interpreted as  $p \approx -\mathbf{n} \cdot \mathbf{F}$ .

### Normal Stress

The total stress on the boundary is set equal to a stress vector of magnitude,  $f_0$ , oriented in the negative normal direction:

$$(-p\mathbf{I} + \eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T))\mathbf{n} = -f_0\mathbf{n}$$

This implies that the total stress in the tangential direction is zero. This boundary condition implicitly sets a constraint on the pressure that for 2D flows can be written

$$p = 2\eta\frac{\partial u_n}{\partial n} + f_0 \quad (6-5)$$

If  $\partial u_n/\partial n$  is small, Equation 6-5 can be interpreted as  $p \approx f_0$ .

### Normal Stress, Normal Flow

In addition to the stress condition set in the Normal stress condition, this condition also prescribes that there must be no tangential velocities on the boundary:

$$(-p\mathbf{I} + \eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T))\mathbf{n} = -f_0\mathbf{n}, \quad \mathbf{t} \cdot \mathbf{u} = 0$$

Also this boundary condition implicitly sets a constraint on the pressure that for 2D flows can be written

$$p = 2\eta\frac{\partial u_n}{\partial n} + f_0 \quad (6-6)$$

If  $\partial u_n/\partial n$  is small, Equation 6-6 can be interpreted as  $p \approx f_0$ .

## Point Settings

If it is not possible to specify the pressure level using a boundary condition, the pressure must be set in some other way, for example, by specifying a fixed pressure at a point. You find a dialog box for **Point Settings** on the **Physics** menu.

## Numerical Stability—Stabilization Techniques

The momentum equations (see Equation 6-1) are (nonlinear) convection-diffusion equations. As described in Chapter 17, “Stabilization Techniques,” such equations are unstable if discretized using the Galerkin finite element method. Stabilized finite element methods are therefore necessary in order to obtain physical solutions. You control the stabilization settings from the **Stabilization** page in the **Subdomain Settings**, which appears in Figure 6-2.

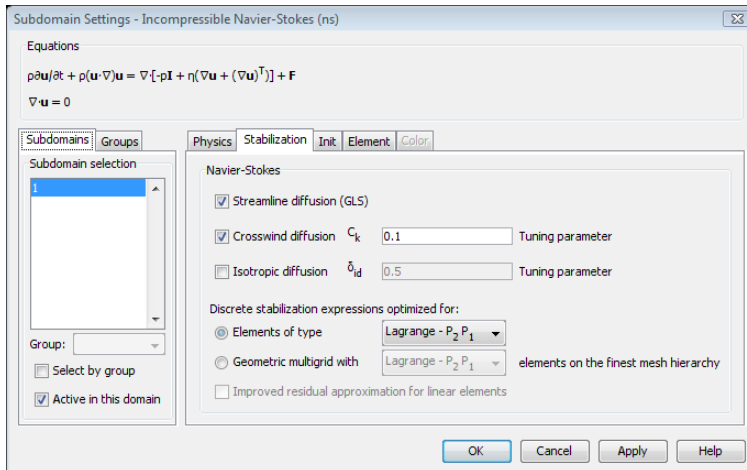


Figure 6-2: The Stabilization page of the Subdomain Settings dialog box.

There are three types of stabilization available in the Incompressible Navier-Stokes application mode:

- Streamline diffusion (GLS)
- Crosswind diffusion
- Isotropic diffusion

For optimal functionality, the exact weak formulations and constants of GLS and crosswind diffusion depend on the order of the basis functions (elements). To control

this, there are two option buttons, each with its corresponding list, at the bottom of the **Stabilization** page shown in Figure 6-2. In most cases, the model utilizes only one element type. If this is the case, select this type from the **Elements of type** list. This is the default setting for all dimensions but 3D.

However, when using the geometric multigrid (GMG) solver/preconditioner, the equations can be discretized with different elements on the different multigrid hierarchies. Then select from the element type that is used on the finest mesh hierarchy from the **Geometric multigrid with** list. This is the default setting for 3D because the default solver for in 3D is BiCGStab with geometric multigrid as the preconditioner. The resulting weak expressions then take forms adapted to multigrid hierarchies created in a way equivalent to the hierarchy generation method that you get when you choose **Lower element order first (any)** from the **Hierarchy generation method** list in the **Linear System Solver Settings** dialog box (see “Constructing a Multigrid Hierarchy” on page 558 of the *COMSOL Multiphysics Reference Guide*).

The values of constants of the GLS and crosswind diffusion follow Ref. 2 and Ref. 3 and cannot be changed without performance losses.

Streamline diffusion and crosswind diffusion are functional only for the element types available in the lists on the **Stabilization** page.

### **STREAMLINE DIFFUSION (GLS)**

For strongly coupled system of equations, the streamline diffusion must be applied to the whole system of equations, not only to each equation separately. These ideas were first explored by Hughes and Mallet (Ref. 4) and were later extended to Galerkin least-squares (GLS) applied to the Navier-Stokes equations (Ref. 5) which is the form that COMSOL supports. The time scale tensor is the diagonal tensor presented in Ref. 6.

Streamline diffusion is active per default since it is necessary when the flow is dominated by convection.

The unstabilized incompressible Navier-Stokes equations are subject the Babuska-Brezzi condition which states that the basis functions for the pressure must be of lower order than the basis functions for the velocity. If the incompressible Navier-Stokes equations are stabilized by GLS, then it is possible to use equal order interpolation. Hence, streamline diffusion is necessary when using linear Lagrange elements. This applies also if the model is solved using geometric multigrid (GMG) as a solver or a preconditioner, and at least one multigrid hierarchy uses linear Lagrange elements.

### CROSSWIND DIFFUSION

Crosswind diffusion can also be formulated for system of equations and when applied to the Navier-Stokes equations it becomes a shock capturing operator. COMSOL supports the formulation in Ref. 5 with shock capturing viscosity taken from Ref. 7.

Incompressible flows do not contain shock waves, but crosswind diffusion is still useful to introduce extra diffusion in sharp boundary layers and shear layers that otherwise would require a very dense mesh to resolve.

The tuning parameter,  $C_k$ , controls the amount of crosswind diffusion introduced in the model. The recommended range for low Mach number flows and incompressible flows is  $0 < C_k < 0.5$  ( $C_k=0$  means no diffusion at all). The value must not be neither space, nor time dependent.

Crosswind diffusion is active per default since it makes it easier to obtain a solution even if the mesh is not perfect. It is, however, advisable to investigate the influence  $C_k$ .

Crosswind diffusion also has the effect that iterative solvers can use inexpensive presmoothers such as SSOR (see “Solver Settings” on page 149). If you deactivate crosswind diffusion, you must change all applications of SSOR to Vanka (see “Preconditioners for the Iterative Solvers” on page 553 in the *COMSOL Multiphysics Reference Guide*).

### ISOTROPIC DIFFUSION

Isotropic diffusion adds diffusion to the momentum equations of the Navier-Stokes equations in the same way as described in “Isotropic Diffusion” on page 500. The continuity equation gain no extra stability.

### IMPROVED RESIDUAL APPROXIMATION FOR LINEAR ELEMENTS

Both GLS and crosswind diffusion evaluate the residuals of the Navier-Stokes equations to determine the amount of diffusion. The residuals of the momentum equations contain second order derivatives of the velocity which, if linear elements are used, are evaluated to zero. The impact of this is small if the flow is dominated by convection where

$$\partial \mathbf{u} / \partial t + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p \quad (6-7)$$

is a good approximation of the momentum equations. Equation 6-7 is not a good approximation for flows dominated by viscous effects, however, and if the mesh is coarse, the result can be that the pressure field contains small oscillations.

The oscillations can be removed by refining the mesh. An alternative is to select the **Improved Residual Approximation for Linear Elements** check box shown in Figure 6-2. The viscous terms are then reconstructed by polynomial-preserving recovery (see “Using Special Operators” on page 163 of the *COMSOL Multiphysics User’s Guide*). The polynomial-preserving recovery makes the equation system more complicated and thereby more expensive to solve.

The improved residuals can be useful if the Navier-Stokes equations are part of a multiphysics problem and you cannot afford to refine the mesh.

### Corner Smoothing

You find the **Corner smoothing** property in the **Application Mode Properties** dialog box. It can be a useful property when the model contains walls with slip conditions, as described below.

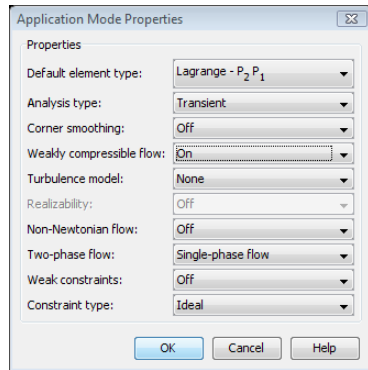


Figure 6-3: The Application Mode Properties dialog box where Corner smoothing can be turned on and off.

Consider the situation sketched in Figure 6-4. At the point where the boundaries  $\Gamma_1$  and  $\Gamma_2$  intersect, there are two boundary normals, one for  $\Gamma_1$  and one for  $\Gamma_2$ . These two normals are denoted  $\mathbf{n}_{\Gamma_1}$  and  $\mathbf{n}_{\Gamma_2}$  in Figure 6-4. If the boundaries now both have no-penetration condition, there are two Dirichlet conditions at the point of intersection, namely

$$\mathbf{n}_{\Gamma_1} \cdot \mathbf{u} = 0 \quad (6-8)$$

and

$$\mathbf{n}_{\Gamma_2} \cdot \mathbf{u} = 0 \quad (6-9)$$

The only way that both Equation 6-8 and Equation 6-9 can be fulfilled is if  $\mathbf{u} \equiv \mathbf{0}$  at the point of intersection. This is not always the expected solution, however.

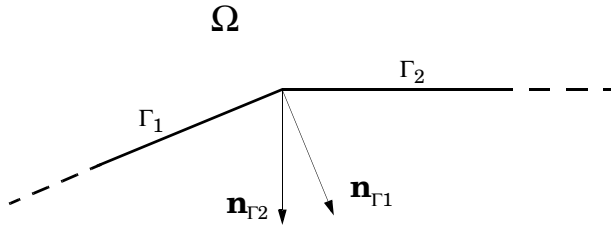


Figure 6-4: Intersection between the boundaries  $\Gamma_1$  and  $\Gamma_2$ .  $\mathbf{n}_{\Gamma_1}$  and  $\mathbf{n}_{\Gamma_2}$  are the boundary normals prescribed by  $\Gamma_1$  and  $\Gamma_2$  respectively.  $\Omega$  is the computational domain.

When corner smoothing is activated, any Dirichlet condition  $d(\mathbf{n}) = 0$  is replaced by  $d(\mathbf{n}_w) = 0$ , where  $\mathbf{n}_w$  is a vector of dependent boundary variables whose solution in each point is the average of all normals in that point. In the current example, equations 6-8 and 6-9 are replaced with

$$\mathbf{n}_w \cdot \mathbf{u} = 0 \quad (6-10)$$

and  $\mathbf{n}_w$  has the solution  $\mathbf{n}_w = (\mathbf{n}_{\Gamma_1} + \mathbf{n}_{\Gamma_2})/2$ . Equation 6-10 can then be satisfied for  $\mathbf{u} \neq \mathbf{0}$ .

### Application Mode Variables

A number of variables and physical quantities are available for postprocessing and for use in equations and boundary conditions. They appear in the following table (where  $x_i$  denotes the various space coordinate directions and, in the **Type** column, V represents vertices (points, B represents boundaries, and S represents subdomains):

NAME	TYPE	DESCRIPTION	EXPRESSION
$u, v, (w)$	B S V	$x_i$ velocity	$u, v, w$
$p$	B S	Pressure	$p$
$U$	B S	Velocity field	$\sqrt{\sum_i (u_i)^2}$
$V$	S	Vorticity	$\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}$ (2D)



NAME	TYPE	DESCRIPTION	EXPRESSION
V <sub>xi</sub>	S	Vorticity	$x_i$ components of $\nabla \times \mathbf{u}$ (3D)
rho	S	Density	$\rho$
eta	S	Dynamic viscosity	$\eta$
F <sub>xi</sub>	S	Volume force, $x_i$ dir.	$x_i$ components of $\mathbf{F}$
K <sub>xi</sub>	B	Viscous force per area, $x_i$ component	$\sum_j n_j \eta \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$
T <sub>xi</sub>	B	Total force per area, $x_i$ component	$\sum_j n_j \left[ -p \delta_{ij} + \eta \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right]$
cellRe	S	Cell Reynolds number	$\frac{\rho  \mathbf{u}  h}{\eta}$
res <sub>ui_c</sub>	S	Equation residual, $u_i$ component	$\rho (\mathbf{u} \cdot \nabla) \mathbf{u} + \nabla p - \mathbf{F}$ $-\nabla \cdot [\eta (\nabla \mathbf{u} + (\nabla \mathbf{u})^T)]$ $(u_i \text{ component})$
res <sub>sc_ui_c</sub>	S	Shock capturing residual, $u_i$ component	$\rho (\mathbf{u} \cdot \nabla) \mathbf{u} + \nabla p - \mathbf{F}$ $(u_i \text{ component})$
beta <sub>xi</sub>	S	Convective field, $x_i$ component	$\rho u_i$
Dm	S	Mean diffusion coefficient	$\eta$
da	S	Total time-scale factor	$\rho$

**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: ns), for example, v<sub>ns</sub>. (This does not apply to the dependent variables for the velocities and pressure.)

### *Solver Settings*

#### **LINEAR SYSTEM SOLVER**

The recommended solver type for small and medium-sized fluid-flow problems is a direct solver, and the PARDISO direct solver is the default solver for 1D and 2D models. It is not as robust as UMFPACK or SPOOLES but is more memory efficient.

3D problems are often too large to solve using a direct solver. The default solver is BiCGStab with the geometric multigrid (GMG) preconditioner. The GMG preconditioner uses the SSOR preconditioner/smoothen as the presmoother and postsmoother. It is possible to use SSOR thanks to the GLS streamline diffusion and to the crosswind diffusion (see “Numerical Stability—Stabilization Techniques” on page 144). If you deactivate any of these stabilization settings, then you must replace the SSOR smoothers by the Vanka preconditioner/smoothen (see “The Vanka Algorithm” on page 568 in the *COMSOL Multiphysics Reference Guide*).

For more information about the solvers, see Chapter 6, “Solving the Model,” on page 377 in the *COMSOL Multiphysics User’s Guide*.

### TRANSIENT SOLVER SETTINGS

The transient Incompressible Navier-Stokes application mode uses per default the generalized- $\alpha$  time-stepping method (see “The Time-Dependent Solver” on page 391 of the *COMSOL Multiphysics User’s Guide*) with manual time step control.

The time step for the transient Incompressible Navier-Stokes application mode is calculated from the CFL number:

$$\text{CFL} = \frac{|\mathbf{u}|\Delta t}{h}$$

The Navier-Stokes application mode contains a variable `dt_cfl_gn_app` that corresponds to a CFL number equal to one.  $n$  is the index of the geometry on which the application mode is active and *app* is the application mode name. As an example, assume that the application mode name is `ns` and that the application mode is active on geometry 2. The time step variable is then `dt_cfl_g2_ns`.

`dt_cfl_gn_app` is defined by

$$\text{dt\_cfl\_gn\_app} = \min_{\Omega_e} \left( \frac{h}{\max(|\mathbf{u}|, f_{vv} \mathbf{u}_{\text{diff}})} \right)$$

where  $\Omega_e$  is the set of all mesh cells in geometry  $n$  on subdomains where the Navier-Stokes is active.  $\mathbf{u}_{\text{diff}}$  is a diffusive velocity scale equal to  $\eta/(\rho h)$ . The argument  $f_{vv} \mathbf{u}_{\text{diff}}$  is necessary to prevent the time step from becoming too large when  $|\mathbf{u}|$  is small. Because  $\mathbf{u}_{\text{diff}}$  is often extremely small, the algorithm multiplies it by a factor,  $f_{vv}$ , which per default is 10.

A necessary condition for explicit time stepping techniques to be stable is that the CFL number is smaller than one. COMSOL uses implicit schemes which are stable for any

$\Delta t$ . Large  $\Delta t$  has the advantage that the computation can be completed in few time steps. There are however two drawbacks:

- The solver introduces numerical diffusion that increases with  $\Delta t$ . In other words, using large  $\Delta t$  means that the result becomes less accurate compared to using small  $\Delta t$ .
- The equation system becomes harder to solve for large  $\Delta t$ . That is, more computational work is required to complete each time step.

A compromise between efficiency and accuracy is to use a  $\Delta t$  such that  $2 \leq \text{CFL} \leq 10$ . The default is  $\Delta t = 5 \cdot dt\_cfl\_app$ , and you find this setting on the **Time Stepping** page in the **Solver Parameters** dialog box.

Use the following guidelines if you want to modify the time step:

- The time step can be increased if you expect the model to feature only very slow transients.
- Decrease the time step if the solver repeatedly fails to solve the nonlinear system (see “The Time-Dependent Solver Log” on page 452 of the *COMSOL Multiphysics User’s Guide*).
- In situations where the viscosity is high,  $f_{vv}=10$  can result in very small time steps. The factor  $f_{vv}$ , the viscous velocity factor, is available as an application scalar variable named `visc_vel_fact_app` and you can set it to any suitable value in the **Application Scalar Variables** dialog box (choose **Scalar Variables** from the **Physics** menu). A lower value of  $f_{vv}$  results in a larger maximum time step length.
- Time steps based on the CFL number do not increase if the model reaches a steady state. Free time stepping can be used if the model is well resolved. The time steps then increase when the simulation approaches steady state. You can change to free time stepping by selecting **Free** from the **Time steps taken by solver** list on the **Time Stepping** page in the **Solver Parameters** dialog box.

In some models the transient behavior is not guided by the Navier-Stokes equations, and the CFL condition can be either too restrictive or too lax. If this is the case, go to the **Time Stepping** page in the **Solver Parameters** dialog box. Then change the **Time steps taken by solver** to **Free** or edit the **Time step** expression to take the time scale of the guiding physics into account.

#### ADVANCED SOLVER SETTINGS

For a flow that is parallel to a coordinate axis, the automatic scaling feature in COMSOL Multiphysics does not work if you use the nonlinear stationary solver. For

such cases, turn off the scaling feature or use manual scaling; see “Scaling of Variables and Equations” on page 529 of the *COMSOL Multiphysics Reference Guide*. The problem occurs when one solution component is on average zero.

### *Khan and Richardson Force for Particle Tracing*

---

The *Khan and Richardson force* is available for particle tracing plots in the Incompressible Navier-Stokes application mode and other application modes for fluid dynamics in the COMSOL Multiphysics products.

#### **BACKGROUND**

The force expression that the software uses is derived partly using experimental results, and it is valid for a wide range of Reynolds numbers, stretching from creeping flow toward the turbulent regime (Ref. 8).

The following equation describes the total force that a fluid exerts on an immersed spherical particle:

$$F = \pi r_p^2 \rho (\bar{u} - \bar{u}_p)^2 (1.84(\text{Re}_p)^{-0.31} + 0.293(\text{Re}_p)^{0.06})^{3.45}$$

where the definition of the particle Reynolds number,  $\text{Re}_p$ , is

$$\text{Re}_p = (|\bar{u} - \bar{u}_p| 2r_p \rho) / \eta.$$

The orientation of a given force component (for example, positive or negative  $x$ -component) is determined by the sign of the corresponding component in the vector difference  $\bar{u} - \bar{u}_p$ , because this determines whether the fluid is accelerating or slowing down the particle in that direction.

#### **USING THE KHAN AND RICHARDSON FORCE FOR PARTICLE TRACING**

The Khan and Richardson force is the default selection in the **Predefined forces** list on the **Particle Tracing** tab.

There is one parameter in this force expression: the particle radius  $r_p$ , which you define by clicking the **Parameters** button. Table 6-1 shows the default values for this parameter:

TABLE 6-1: PARAMETER FOR THE KHAN AND RICHARDSON FORCE

PARAMETER NAME IN EQUATION	DESCRIPTION	DEFAULT VALUE	DEFAULT VARIABLE NAME
$r_p$	particle radius	$10^{-4}$ m	partr

The particle mass appears in the force equation, and you enter its value in the **Mass** edit field. The default value for the particle mass is  $m_p = (4\pi/3) \cdot 10^{-9}$  kg, which is the mass of a particle with the same density as water and a radius of  $10^{-4}$  m, which is the default radius.

---

**Note:** The default settings work for models that use SI units.

---

### *References*

---

1. P.M. Gresho and R.L. Sani, *Incompressible Flow and the Finite Element Method, Volume 2: Isothermal Laminar Flow*, John Wiley and Sons, LTD, 2000.
2. I. Harari and T.J.R. Hughes, “What are  $C$  and  $h$ ? Inequalities for the analysis and design of finite element methods,” *Comp. Meth. Appl. Mech. Engrg*, vol. 97, pp. 157–192, 1992.
3. Y. Bazilevs, V.M. Calo, T.E. Tezduyar, and T.J.R. Hughes, “ $\text{YZ}\beta$  discontinuity capturing for advection-dominated processes with application to arterial drug delivery,” *Int.J.Num. Meth. Fluids*, vol. 54, pp. 593–608, 2007.
4. T.J.R. Hughes and M. Mallet, “A new finite element formulation for computational fluid dynamics: III. The Generalized Streamline Operator for Multidimensional Advective-Diffusive System,” *Comp. Meth. Appl. Mech. Engrg*, vol. 58, pp. 305–328, 1986.
5. G. Hauke and T.J.R. Hughes, “A unified approach to compressible and incompressible flows” *Comp. Meth. Appl. Mech. Engrg*, vol. 113, pp. 389–395, 1994.
6. G. Hauke, “Simple stabilizing matrices for the computation of compressible flows in primitive variables,” *Comp. Meth. Appl. Mech. Engrg*, vol. 190, pp. 6881–6893, 2001.
7. A. Soulaïmani and M. Fortin, “Finite element solution of compressible viscous flows using conservative variables,” *Comp. Meth. Appl. Mech. Engrg*, vol. 118, pp. 319–350, 2001.
8. J.M. Coulson and J.F. Richardson, “Particle Technology and Separation Processes,” *Chemical Engineering*, vol. 2, Butterworth-Heinemann.

# Example—Steady Incompressible Flow

## *Introduction*

---

This model examines the physics of plane, incompressible, and steady flow: flow over a backward step in the absence of external forces. This is a common benchmark problem in CFD. There is no known exact solution, but experimental data has been published (see Ref. 1) making it possible to check the accuracy of the FEM solution. The model includes analyses using both regular triangular meshes and mapped meshes, comparing the solutions for various mesh densities.

## *Model Definition*

---

Fluid enters from the left side with a parabolic velocity profile, passes over a step, and leaves through the right boundary (Figure 6-5 shows the model geometry).



*Figure 6-5: The backward geometry.*

The model computes the fluid's velocity components  $\mathbf{u} = (u, v)$  in the  $x$  and  $y$  directions and its pressure  $p$  in the region defined by the geometry in the preceding figure. The PDE model for this application uses the stationary incompressible Navier-Stokes equations

$$\begin{cases} -\eta \nabla^2 \mathbf{u} + \rho(\mathbf{u} \cdot \nabla) \mathbf{u} + \nabla p = \mathbf{F} \\ \nabla \cdot \mathbf{u} = 0 \end{cases}$$

where the quantities are (using SI units):

- Dynamic viscosity,  $\eta = 1.79 \cdot 10^{-5}$  Pa·s
- Density,  $\rho = 1.23$  kg/m<sup>3</sup>
- A force field,  $\mathbf{F}$ , absent in this model

The first equation is the balance of momentum from Newton's second law. The other relationship is the equation of continuity, where zero on the right-hand side states that the fluid is incompressible.

The shape of the flow pattern depends only on the Reynolds number.

In this model, you choose boundary conditions so that the average velocity at the inlet is  $V_{\text{mean}} = 0.544$  m/s. To obtain a corresponding parabolic velocity profile, set  $(u, v) = (6V_{\text{mean}}s(1-s), 0)$ , where  $s$  is a boundary parameterization variable that runs from 0 to 1 along the boundary. The fluid is always stationary at the walls, so  $(u, v) = (0, 0)$  is the appropriate boundary condition. At the exit boundary, assume a constant static pressure  $p = 0$ .

For such a fluid flow you can expect a velocity field with a boundary layer of thickness approximately equal to  $1/\sqrt{\text{Re}}$  at the walls. To resolve this steep solution gradient you need a few rows of elements across the layer. For a flow with a large Reynolds number, elements in the interior of the channel can be much larger than those near the walls.

#### USING THE INCOMPRESSIBLE NAVIER-STOKES APPLICATION MODE

Start by setting up a model and solve this problem on a fixed isotropic mesh. The given input data correspond to a Reynolds number of

$$\text{Re} = \frac{0.544 \cdot 2 \cdot 0.0052 \cdot 1.23}{1.79 \cdot 10^{-5}} \approx 389$$

Even though you are working with a steady flow model, it needs initial conditions because the incompressible Navier-Stokes equations are nonlinear. To achieve a numerical solution, the nonlinear solver solves the equations iteratively.

#### Results

---

Figure 6-6 shows the velocity field in a streamline plot. Behind the step you can identify a recirculation region that expands with increasing  $\text{Re}$ . The distance from the step to the *stagnation point*, where the flow reattaches to the lower wall, is the *reattachment length*. A dimensionless number quantifying the recirculation region is

the reattachment length divided by the step height. According to experimental data, this quotient is approximately 7.93 at a Reynolds number of 389.

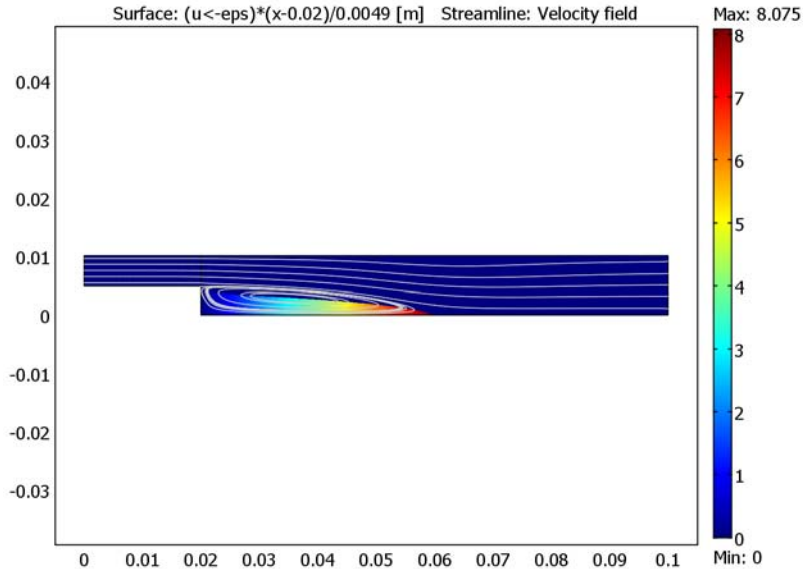


Figure 6-6: A streamline plot of the velocity field.

For a high enough  $Re$ , the eddy crosses the outflow boundary, which means that there is an inflow through the assumed outflow boundary. This finding calls for the inclusion of more of the downstream region in the computational domain to get a reliable solution. That fact explains why the computational domain in this model is slightly longer than the one in Ref. 2.

The Incompressible Navier-Stokes application mode uses Lagrange p2-p1 elements to stabilize the pressure. Thus 2nd-order Lagrange elements model the velocity components while linear elements model the pressure. The default element settings in this application mode always provide one order higher Lagrange elements for the velocity components than for the pressure. Stabilizing the solution using streamline diffusion is not recommended for the Navier-Stokes equations except at higher Reynolds numbers.

Four mesh cases were considered for this model: a homogeneous structured mesh, a homogeneous unstructured mesh, and two inhomogeneous mesh cases, where local



mesh refinements provide improved accuracy around the point where the step is taken. The following plots show examples of the mesh cases:

Case 1: Homogeneous structured mesh



Case 2: Homogeneous unstructured mesh



Case 3: Structured mesh



Case 4: Unstructured mesh



To compare the results using these mesh cases, we solved the model for varying mesh densities for all four mesh cases. The graph in Figure 6-7 shows the resulting

reattachment length divided by the step height as a function of the degrees of freedom for the different mesh cases:

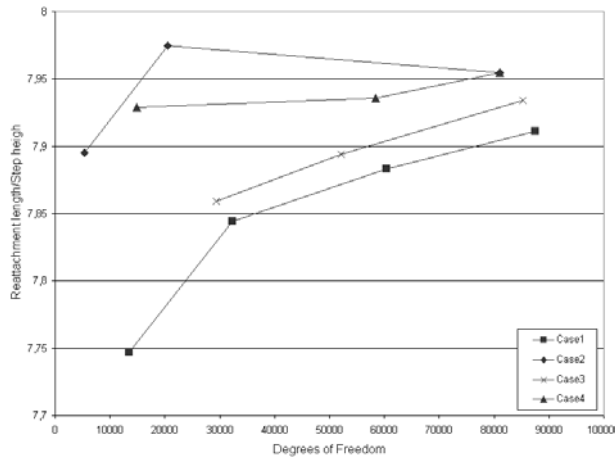


Figure 6-7: Reattachment length divided by step height as functions of the degrees of freedom for Case 1 (squares), Case 2 (diamonds), Case 3 (x-marks), and Case 4 (triangles).

All solutions give reasonable results. Comparing results for structured and unstructured meshes and different numbers of degrees of freedom (DOFs), it is evident that the unstructured mesh cases give more accurate and more stable results even with a lower number of DOFs. However, in the limit of an infinite number of DOFs, the results for all mesh cases converge to the experimental result of 7.93.

The number of mesh elements is related to the number of DOFs. However, the relation is not the same for structured and unstructured meshes. The same number of mesh elements create more DOFs for a structured mesh than for an unstructured mesh, simply because of the rectangular shape of the mesh elements.

For the same number of DOFs the solution time is longer for a structured mesh than for an unstructured mesh. This is due to the structured mesh having a stronger coupled system, which results in system matrices that are less sparse.

## References

1. P.M. Gresho and R.L. Sani, *Incompressible Flow and the Finite Element Method*, Volume 1 & 2, John Wiley & Sons, New York, 2000.

2. A. Rose and B. Simpson: “Laminar, Constant-Temperature Flow Over a Backward Facing Step,” *1st NAFEMS Workbook of CFD Examples*, Glasgow, UK, 2000.

---

**Model Library path:** COMSOL\_Multiphysics/Fluid\_Dynamics/backstep

---

---

**Model Library path:** COMSOL\_Multiphysics/Fluid\_Dynamics/backstep\_quad

---

### *Modeling Using the Graphical User Interface*

---

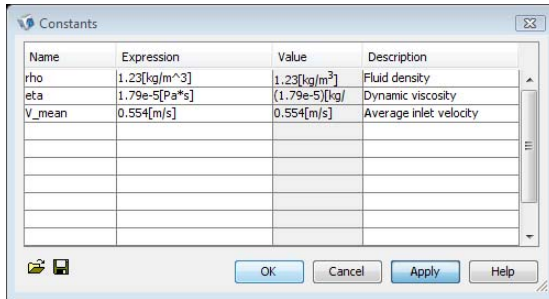
#### **MODEL NAVIGATOR**

- 1 Go to the **Model Navigator** and select **2D** in the **Space dimension** list.
- 2 In the list of application modes open the **COMSOL Multiphysics>Fluid Dynamics** folder and then the **Incompressible Navier-Stokes** folder. Select **Steady-state analysis**.
- 3 Make sure that **Lagrange - P<sub>2</sub> P<sub>1</sub>** is the element type in the **Element** list.
- 4 Click **OK**.

#### **OPTIONS AND SETTINGS**

- 1 To parameterize the model, go to the **Options** menu and choose **Constants**.
- 2 Make the following entries in the **Constants** dialog box to represent the fluid properties and velocity (the descriptions are optional):

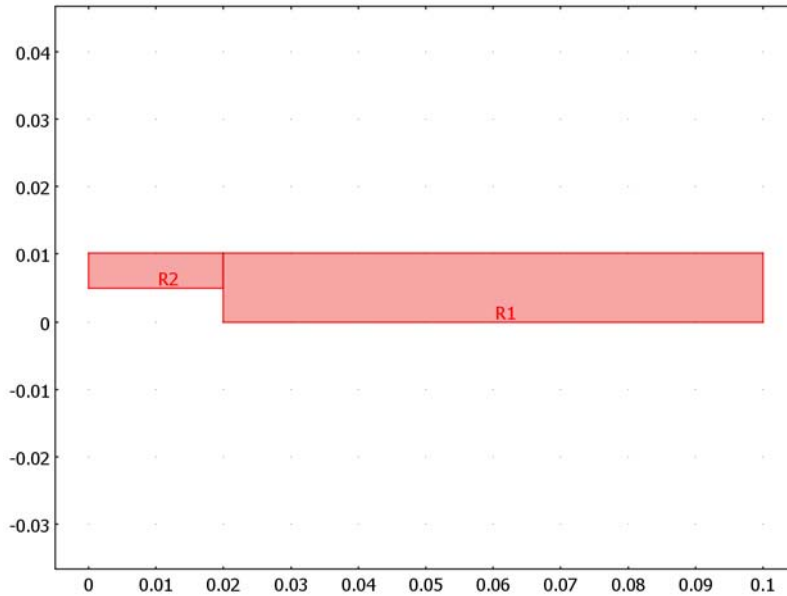
NAME	EXPRESSION	DESCRIPTION
rho	1.23[kg/m <sup>3</sup> ]	Fluid density
eta	1.79e-5[Pa*s]	Dynamic viscosity
V_mean	0.554[m/s]	Average inlet velocity



3 Click **OK** to close the dialog box.

### GEOMETRY MODELING

- 1 Shift-click the **Rectangle/Square** button to specify a rectangle.
- 2 In the **Rectangle** dialog box, type 0.08 in the **Width** edit field and 0.0101 in the **Height** edit field. In the **Position** area, type 0.02 in the **x** edit field (keep the **Base** setting **Corner**).
- 3 Click **OK**.
- 4 Shift-click the **Rectangle/Square** button to specify another rectangle.
- 5 In the **Rectangle** dialog box, type 0.02 in the **Width** edit field and 0.0052 in the **Height** edit field. In the **Position** area, type 0.0049 in the **y** edit field (again, keep the **Base** setting **Corner**).
- 6 Click **OK**.
- 7 Click the **Zoom Extents** button on the Main toolbar.



## PHYSICS SETTINGS

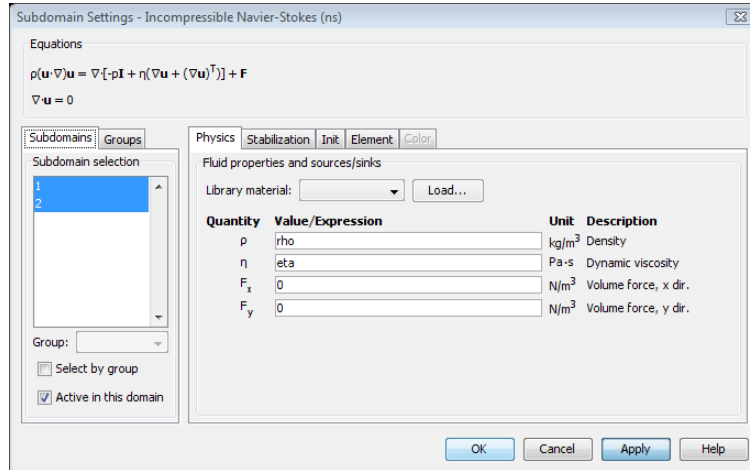
### *Subdomain Settings*

With appropriate scaling it is possible to set the fluid density  $\rho = 1$  and the viscosity  $\eta = 1/\text{Re}$ . If you model in this way, the Reynolds number contains information about fluid density together with details on length scale, velocity scale, and viscosity. For such a model, type  $1/\text{Re}$  in the **Dynamic viscosity** edit field in the **Subdomain Settings** dialog box. Be aware, though, that you must reverse the scaling before interpreting the results quantitatively.

This model, however, uses actual fluid properties specified in SI units.

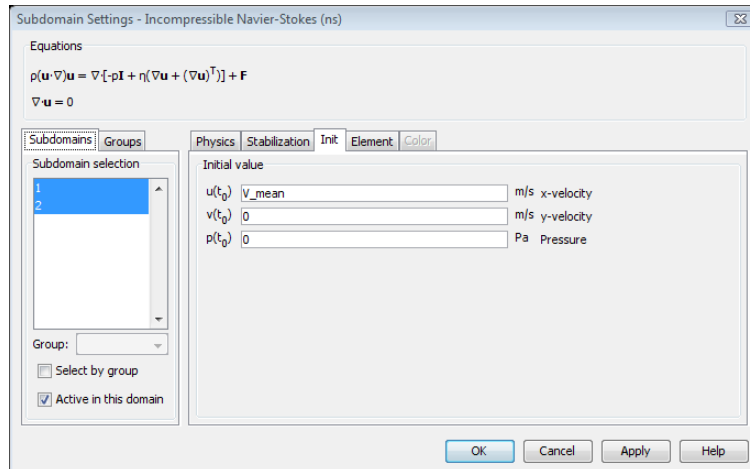
- 1 From the **Physics** menu choose **Subdomain Settings** to launch the **Subdomain Settings** dialog box.
- 2 Select Subdomains 1 and 2.
- 3 In the **Density** edit field type rho.

4 In the **Dynamic viscosity** edit field type eta.



The nonlinear solver also requires an initial value. With a low Reynolds number a simple constant expression is sufficient:

- 5 With both subdomains still selected, click the **Init** tab.
- 6 Enter the initial value  $V_{\text{mean}}$  in the edit field for  $\mathbf{u}(\mathbf{t}_0)$ ; this value corresponds to a uniform  $x$ -velocity of magnitude  $V_{\text{mean}}$  throughout the domain. Leave the initial values for the  $y$ -velocity and the pressure at their default zero values.



7 Click **OK**.

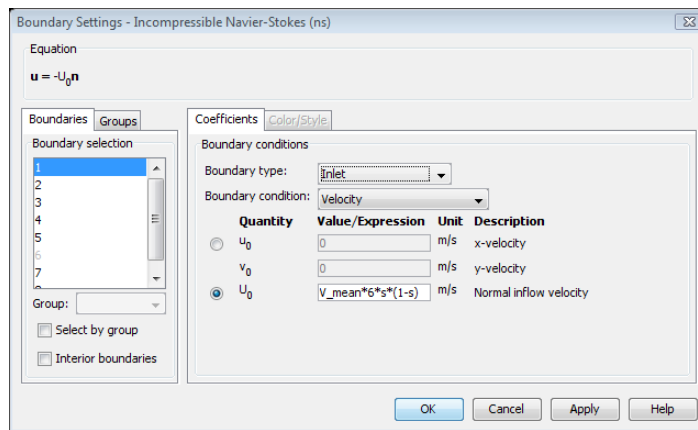
### Boundary Conditions

The relevant boundary conditions for the model are:

SETTINGS	BOUNDARY 1	BOUNDARIES 2–5, 7	BOUNDARY 8
Boundary type	Inlet	Wall	Outlet
Boundary condition	Velocity	No slip	Pressure, no viscous stress
$U_0$	$V\_mean*6*s*(1-s)$		
$P_0$			0

The no-slip condition is the default setting, so you only need to modify the settings for Boundaries 1 and 8.

- 1 From the **Physics** menu, choose **Boundary Settings**.
- 2 Select Boundary 1.
- 3 In the **Boundary type** list select **Inlet**.
- 4 In the  $U_0$  edit field type  $V\_mean*6*s*(1-s)$ .



- 5 Select Boundary 8.
- 6 In the **Boundary type** list select **Outlet**.
- 7 Click **OK**.

### MESH GENERATION

Because the Navier-Stokes equations are computationally difficult, it is important to use an appropriate mesh. If the mesh is too coarse, the solution might not converge at all or errors might be large. Conversely, if the mesh is too fine, the solution time for

the nonlinear system of equations might be unnecessarily long. In this case, solve the model with two different types of mesh, first using a mapped mesh and later using a conventional unstructured triangular mesh.

*Mesh Case 1—Mapped mesh*

- 1 From the **Mesh** menu, choose **Mapped Mesh Parameters**.
- 2 In the **Mapped Mesh Parameters** dialog box, click the **Boundary** tab.
- 3 Specify the numbers of mesh elements according to the following table by first selecting the appropriate boundaries in the **Boundary selection** list, then selecting the **Constrained edge element distribution** check box, and finally entering the number of elements in the **Number of edge elements** edit field.

SETTINGS	BOUNDARIES 1-4, 6	BOUNDARIES 5, 7
Number of mesh elements	34	62

- 4 Click **Remesh**, then click **OK**.

*Mesh Case 2—Triangular mesh*

- 1 From the **Mesh** menu, choose **Free Mesh Parameters**.
- 2 On the **Global** page, click the **Custom mesh size** option button, then type  $9e-4$  in the **Maximum element size** edit field.
- 3 Click **Remesh**, then click **OK**.

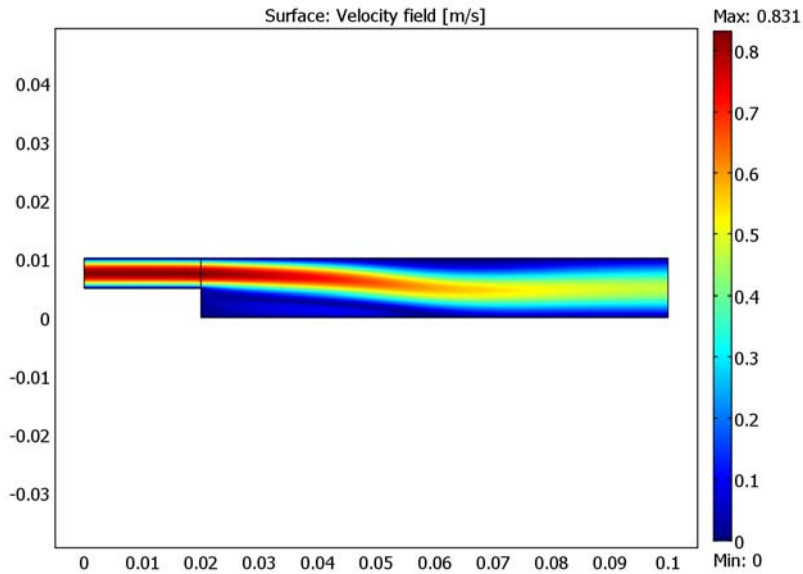
**COMPUTING THE SOLUTION**

Click the **Solve** button on the Main toolbar to start the simulation.



## POSTPROCESSING AND VISUALIZATION

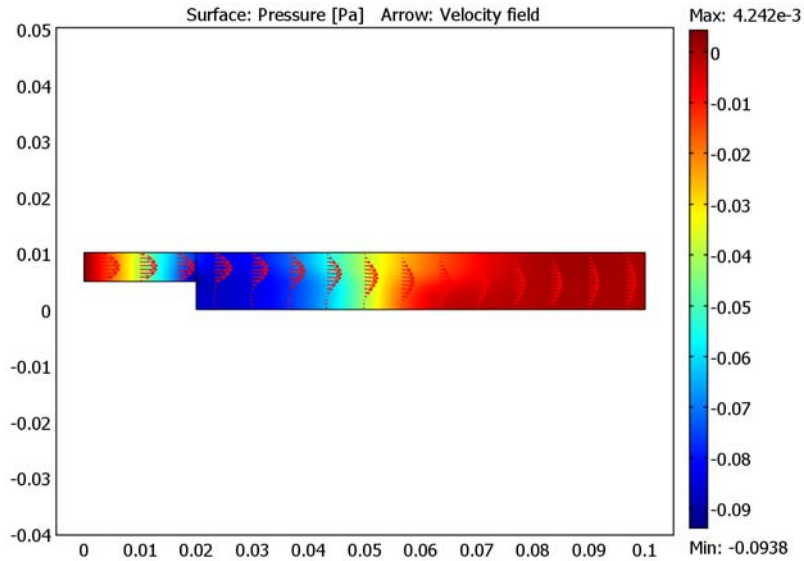
The default visualization displays the magnitude of the velocity field.



To see the velocity field and the pressure field simultaneously, use a combination of arrow and surface plots:

- 1 From the **Postprocessing** menu, choose **Plot Parameters** to open the **Plot Parameters** dialog box.
- 2 Click the **Surface** tab.
- 3 On the **Surface Data** page, select **Incompressible Navier-Stokes (ns)>Pressure** from the **Predefined quantities** list.
- 4 Click the **Arrow** tab.
- 5 Select the **Arrow plot** check box.

6 Click **OK** to view the combined plot.



To reproduce the plot in Figure 6-6, visualizing the recirculation region, use a conditional expression testing for a negative  $x$ -component in the velocity.

- 1 Open the **Plot Parameters** dialog box again. Add a streamline plot and remove the arrows by selecting the **Streamline** check box and clearing the **Arrow** check box on the **General** page.
- 2 Click the **Surface** tab.
- 3 On the **Surface Data** page, type  $(u < -\epsilon) * (x - 0.02) / 0.0049$  in the **Expression** edit field. Clear the **Smooth** check box (the given expression is deliberately discontinuous).
- 4 Click the **Streamline** tab.
- 5 On the **Start Points** page, click the **Specify start point coordinates** button.
- 6 In the **x** edit field type  $\text{linspace}(0.03, 0.03, 9)$  and in the **y** edit field type  $\text{linspace}(5e-4, 95e-4, 9)$ . These values result in nine streamlines, which at  $x = 0.03$  m are equally spaced in the  $y$  direction.
- 7 Click the **Line Color** tab, then click the **Color** button.
- 8 In the **Streamline Color** dialog box, click the lightest gray swatch, then click **OK**.
- 9 Click **OK** to close the **Plot Parameters** dialog box and display the new plot.

# Heat Transfer

This chapter covers heat transfer application modes. It starts with some background on heat transfer. It then reviews specifics of the Conduction application mode and then the Convection and Conduction application mode. It concludes with COMSOL Multiphysics models of three heat-transfer examples taken from a NAFEMS (National Agency for Finite Element Methods and Standards) benchmark collection.

# Heat Transfer Fundamentals

## *What Is Heat Transfer?*

---

From the kitchen toaster to the latest high-performance microprocessor, heat is ubiquitous and of great importance in the engineering world. To optimize thermal performance and reduce costs, engineers and researchers are making use of finite element analysis. Because most material properties are temperature-dependent, the effects of heat enter many other disciplines and drive the requirement for multiphysics modeling.

For instance, both the toaster and the microchip contain electrical conductors that generate thermal energy as electric current passes through them. As these conductors release thermal energy, system temperature increases as does that of the conductors. If the electric conductivity is temperature dependent, it changes accordingly and, in turn, affects the electric field in the conductor. Other examples of multiphysics couplings that involve heat transfer are thermal stresses, thermal-fluid convection, and induction heating.

Heat transfer is defined as the movement of energy due to a temperature difference. It is characterized by the following three mechanisms:

- *Conduction* is heat transfer by diffusion in a stationary medium due to a temperature gradient. The medium can be a solid or a fluid.
- *Convective heat transfer* is when heat is transported by a fluid motion. (Engineers sometimes uses convection to refer to heat transfer between either a hot surface and a cold moving fluid or a cold surface and a hot moving fluid.)
- *Radiation* is heat transfer via electromagnetic waves between two surfaces (A and B) with different temperatures  $T_A$  and  $T_B$ , provided that Surface A is visible to an infinitesimally small observer on Surface B.

---

**Note:** The Heat Transfer Module supports simulations of all three types of heat transfer mechanism, including surface-to-surface and surface-to-ambient radiation.

---

The examples later in this chapter shows transient heat transfer by conduction, convection, and radiation. For an introductory example of a multiphysics coupling of a heat balance to a momentum balance through the Navier-Stokes equations, see “Free

Convection” on page 362 of the *COMSOL Multiphysics Model Library*. In that model the heat flux accounts for transport by convection and conduction. The system consists of heated tubes subjected to a stream of a fluid perpendicular to the main axis of the tubes. The tubes heat the streaming medium as it travels from the bottom to the top of the domain.

### *The Heat Equation*

---

The mathematical model for heat transfer by conduction is the *heat equation*:

$$\rho C_p \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = Q$$

Quickly review the variables and quantities in this equation:

- $T$  is the temperature.
- $\rho$  is the density.
- $C_p$  is the *heat capacity* at constant pressure.
- $k$  is *thermal conductivity*.
- $Q$  is a *heat source* or a *heat sink*.

For a steady-state model, temperature does not change with time, and the first term containing  $\rho$  and  $C$  vanishes.

If the thermal conductivity is anisotropic,  $k$  becomes the thermal conductivity tensor:

$$k = \begin{bmatrix} k_{xx} & k_{xy} & k_{xz} \\ k_{yx} & k_{yy} & k_{yz} \\ k_{zx} & k_{zy} & k_{zz} \end{bmatrix}$$

To model heat conduction and convection through a fluid, the heat equation also includes a convective term. COMSOL Multiphysics represents this formulation in the Convection and Conduction application mode as:

$$\rho C_p \frac{\partial T}{\partial t} + \rho C_p \mathbf{u} \nabla \cdot T = \nabla \cdot (k \nabla T) + Q$$

where  $\mathbf{u}$  is the velocity field. This field can either be provided as a mathematical expression of the independent variables or calculated by a coupling to the velocity field from an application mode such as Incompressible Navier-Stokes.

For transport through conduction and convection, a heat flux vector can be approximated by

$$\mathbf{q} = -k\nabla T + \rho C_p T \mathbf{u}$$

where  $\mathbf{q}$  is the heat flux vector. If the heat transfer is by conduction only,  $\mathbf{q}$  is instead determined by

$$\mathbf{q} = -k\nabla T$$

For a detailed discussion of the fundamentals of heat transfer, see Ref. 1.

---

**Note:** *Heat capacity* here refers to the quantity that represents the amount of heat required to change one unit of mass of a substance by one degree. It has units of energy per mass per degree (J/(kg·K) in SI units). This quantity is also called *specific heat* or *specific heat capacity*.

---

# The Conduction Application Mode

This application mode models heat transfer by conduction. It also includes convection and radiation effects around edges and boundaries. The Conduction application mode is suitable for modeling of heat transfer in solids. To start modeling with this application mode, go to the **Model Navigator** and select **Heat Transfer** and then **Conduction**.

## *Variables and Space Dimensions*

---

The Conduction application mode is available in 1D, 2D, and 3D as well as for axisymmetric models using cylindrical coordinates in 1D and 2D. The dependent variable is the temperature,  $T$ .

## *PDE Formulation*

---

The mathematical model for heat transfer by conduction is the following version of the *heat equation*

$$\delta_{ts}\rho C_p \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = Q$$

with the following material properties:

- $\delta_{ts}$  is a *time-scaling coefficient*.
- $\rho$  is the *density*.
- $C_p$  is the *heat capacity*.
- $k$  is the *thermal conductivity* tensor.
- $Q$  is the heat source (or sink).

For a steady-state problem the temperature does not change with time and the first term disappears.

In rare cases you might decide to add domain-specific transversal convection or radiation in 1D planar and axisymmetric models and 2D planar models. Represent this by adding the two terms on the right:

$$\delta_{ts}\rho C_p \frac{\partial T}{\partial t} - \nabla \cdot (k \nabla T) = Q + \frac{h_{\text{trans}}}{dA} (T_{\text{ext}} - T) + \frac{C_{\text{trans}}}{dA} (T_{\text{ambtrans}}^4 - T^4)$$

where:

- $h_{\text{trans}}$  is the *transversal convective heat transfer coefficient*.
- $T_{\text{ext}}$  is the *transversal external temperature*.
- $C_{\text{trans}}$  is a *user-defined constant*.
- $T_{\text{ambtrans}}$  is the *transversal ambient temperature*.
- $dA$  is the thickness in 2D and area in 1D.

---

**Note:** When using transversal convection or radiation, the heat equation must be modified to accommodate the area  $dA$ . For all practical purposes the use of this feature is a rare exception, and the terms  $h_{\text{trans}}$  and  $C_{\text{trans}}$  are usually set to zero.

---

### *Subdomain Settings*

---

The subdomain quantities are:

QUANTITY	VARIABLE	DESCRIPTION
$\delta_{\text{ts}}$	Dts	Time-scaling coefficient
$\rho$	rho	Density
$C_p$	C	Heat capacity
$k$	k	Thermal conductivity
$\underline{k}$		Thermal conductivity tensor
$k_{ij}$	$kxixj$	Thermal conductivity tensor, $x_i x_j$ component
$Q$	Q	Heat source
$h_{\text{trans}}$	htrans	Transversal convective heat transfer coefficient
$T_{\text{ext}}$	Text	Transversal external temperature
$C_{\text{trans}}$	Ctrans	User-defined constant
$T_{\text{ambtrans}}$	Tambtrans	Transversal ambient temperature

**Time-scaling coefficient** This coefficient is normally 1, but you can change the time scale, for example, from seconds to minutes by setting it to 1/60.

**Density** It specifies the material's density. Enter this quantity as mass per volume.



**Heat capacity** The heat capacity  $C$  describes the amount of heat energy required to produce a unit temperature change in a unit mass.

**Thermal conductivity** The thermal conductivity  $k$  describes the relationship between the heat flux vector  $\mathbf{q}$  and the temperature gradient  $\nabla T$  as in

$$\mathbf{q} = -k\nabla T,$$

which is *Fourier's law of heat conduction*. Enter this quantity as power per length and temperature.

**Heat source** The heat source describes heat generation within the domain. Express heating and cooling with positive and negative values, respectively. Enter the quantity  $Q$  as unit power per unit volume ( $\text{W}/\text{m}^3$  in SI units).

### *Boundary Condition Types*

The available boundary conditions are:

BOUNDARY CONDITION	DESCRIPTION
$\mathbf{n} \cdot (k\nabla T) = q_0 + h(T_{\text{inf}} - T) + C_{\text{const}}(T_{\text{amb}}^4 - T^4)$	Heat flux
$\mathbf{n} \cdot (k\nabla T) = 0$	Insulation or symmetry
$T = T_0$	Prescribed temperature
$T = 0$	Zero temperature
$\mathbf{n}_1 \cdot (k_1\nabla T_1) = \frac{k}{d}(T_2 - T_1)$	Thin thermally resistive layer (pair boundaries only)
$\mathbf{n}_2 \cdot (k_2\nabla T_2) = \frac{k}{d}(T_1 - T_2)$	

### HEAT FLUX

$$\mathbf{n} \cdot (k\nabla T) = q_0 + h(T_{\text{inf}} - T) + C_{\text{const}}(T_{\text{amb}}^4 - T^4)$$

COMSOL Multiphysics supplies only one generalized boundary condition for heat flux, but it accounts for general heat flux as well as heat flux from convection and radiation.

The application mode interprets the heat flux  $q_0$  in the direction of the inward normal. It interprets the convection and radiation terms in the direction of the outward normal.

- Specify  $q_0$  to represent a heat flux that enters the domain. For instance, any electric heater is well represented by this condition, and you can omit its geometry. Enter the quantity  $q_0$  as unit power per unit area ( $\text{W}/\text{m}^2$  using SI units). While this is directly applicable to 3D, unit depth and unit area are assumed in 2D and 1D applications, respectively.
- $h(T_{\text{inf}} - T)$  models convective heat transfer with the surrounding environment, where  $h$  is the *heat transfer coefficient* and  $T_{\text{inf}}$  is the ambient bulk temperature. The value of  $h$  depends on the geometry and the ambient flow conditions. For a thorough introduction on how to calculate heat transfer coefficients see Ref. 1.
- $C_{\text{const}}(T_{\text{amb}}^4 - T^4)$  models radiation heat transfer with the surrounding environment.  $T_{\text{amb}}$  is the temperature of the surrounding radiation environment, which might differ from  $T_{\text{inf}}$ .  $C_{\text{const}}$  is the product of the surface emissivity  $\epsilon$  and the Stefan-Boltzmann constant  $\sigma = 5.669 \cdot 10^{-8} \text{ W}/(\text{m}^2 \cdot \text{K}^4)$  (with the same unit as the Stefan-Boltzmann constant):

$$C_{\text{const}} = \epsilon \sigma$$

The surface emissivity is a material property discussed and tabulated in Ref. 1.

---

**Note:** A problem with radiation boundaries is nonlinear, so you must work with absolute (thermodynamical) temperature units. See “Using Units” on page 188 of the *COMSOL Multiphysics User’s Guide*.

---

## INSULATION OR SYMMETRY

$$\mathbf{n} \cdot (k \nabla T) = 0$$

This condition specifies where the domain is well insulated, or it reduces model size by taking advantage of symmetry. Intuitively this equation says that the gradient across the boundary must be zero. For this to be true, the temperature on one side of the boundary must equal the temperature on the other side. Because there is no temperature difference across the boundary, heat cannot transfer across it.

An interesting numerical check for convergence is the numerical evaluation of the above condition along the boundary, something easily accomplished with the

postprocessing features of COMSOL Multiphysics. Another check is to plot the temperature field as a contour plot. Ideally the contour lines should be perpendicular to any insulated boundary.

### **AXIAL SYMMETRY**

This boundary condition is available only for axisymmetric versions of the heat transfer application models. Use it only on the symmetry axis  $r = 0$ .

### **PRESCRIBED TEMPERATURE**

$$T = T_0$$

This boundary prescribes the temperature  $T_0$ . This condition means that the finite element solution returns a solution in which the above condition is either true or minimally approximated.

### **PRESCRIBED ZERO TEMPERATURE**

$$T = 0$$

This boundary specifies a zero boundary temperature.

### **CONTINUITY**

The default setting for interior boundaries is Continuity, which is a special case of the above Heat source/sink condition. In the absence of sources or sinks, that condition becomes  $-\mathbf{n}_1 \cdot (\mathbf{q}_1 - \mathbf{q}_2) = 0$ . This means that the heat flux in the normal direction is continuous across the boundary. This is the default boundary condition on interior boundaries.

The temperature is naturally continuous on an internal boundary following the continuity of the finite element field. Therefore, the Continuity boundary condition is identical to the condition that applies between any two neighboring elements in the mesh. In fact, as long as you have not selected **Enable interior boundaries** in the **Boundary Settings** dialog box, the interior boundaries are not in any way different from any other mesh element boundaries, where the Continuity condition effectively applies.

The Continuity boundary condition is available on interior boundaries and assembly pairs only.

### HEAT FLUX DISCONTINUITY

The Heat flux discontinuity condition is available on interior boundaries and pairs representing borders between parts in an assembly:

$$-\mathbf{n}_1 \cdot \mathbf{q}_1 - \mathbf{n}_2 \cdot \mathbf{q}_2 = q \quad \text{on } \partial\Omega$$

The right-hand side represents a flux discontinuity, or equivalently, a heat source or heat sink depending on sign. The temperature is always continuous due to the continuity of the finite element field. If the right-hand side is zero, this equation specifies continuity also in the normal heat flux.

### THIN THERMALLY RESISTIVE LAYER

Use this boundary condition to model a thin layer of thermally resistive material using these equations:

$$\begin{aligned}\mathbf{n}_1 \cdot (k_1 \nabla T_1) &= \frac{k}{d}(T_2 - T_1) \\ \mathbf{n}_2 \cdot (k_2 \nabla T_2) &= \frac{k}{d}(T_1 - T_2)\end{aligned}$$

The layer has the thickness  $d$  and the thermal conductivity  $k$ . This boundary condition is only available for pairs representing borders between the parts in an assembly.

### *Boundary Settings*

---

You specify the boundary conditions in the **Boundary Settings** dialog box.

QUANTITY	EDIT FIELD	DESCRIPTION
$q_0$	q	Inward heat flux
$h$	h	Convective heat transfer coefficient
$T_{\text{inf}}$	$T_{\text{inf}}$	Ambient bulk temperature
Const	Const	Radiation constant: product of emissivity and Stefan-Boltzmann constant
$T_{\text{amb}}$	$T_{\text{amb}}$	Temperature of the surrounding radiating environment
$T_0$	$T_0$	Prescribed temperature

## Application Mode Variables

The Conduction application mode uses the following variables for domain equations, boundary equations, and postprocessing purposes.

NAME	DOMAIN/ TYPE	DESCRIPTION	EXPRESSION
T	S/B	Temperature	$T$
gradT, Tx <sub>i</sub>	S/V	Temperature gradient	$ \nabla T , \frac{\partial T}{\partial x_i}$
flux	S	Heat flux	$ \underline{-k}\nabla T $
nflux_T	B	Normal heat flux	$\mathbf{n} \cdot (\underline{-k}\nabla T)$
fluxx <sub>i</sub>	V	Heat flux, x <sub>i</sub> component	$\sum_j -k_{ij} \frac{\partial T}{\partial x_j}$
Dts	S	Time-scaling coefficient	$\delta_{ts}$
rho	S	Density	$\rho$
C	S	Heat capacity	$C_p$
k, kx <sub>i</sub> x <sub>j</sub>	S	Thermal conductivity	$k, k_{ij}$
Q	S	Heat source	$Q$
htrans	S	Transversal convective heat transfer coefficient	$h_{trans}$
Text	S	Transversal external temperature	$T_{ext}$
Ctrans	S	User-defined constant	$C_{trans}$
Tambtrans	S	Transversal ambient temperature	$T_{ambtrans}$
T0	B	Prescribed temperature	$T_0$
h	B	Convective heat transfer coefficient	$h$
Tinf	B	Ambient bulk temperature	$T_{inf}$
Const	B	Radiation constant: product of emissivity and Stefan-Boltzmann constant	Const
Tamb	B	Temperature of the surrounding radiating environment	$T_{amb}$

---

**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: `ht`), for example, `flux_ht`. (This does not apply to the dependent variable for the temperature.)

---

---

**Note:** The vector expressions  $V$  are not present in the 1D formulation of the Conduction application mode.

---

# The Convection and Conduction Application Mode

In addition to heat transfer by conduction, the Convection and Conduction application mode includes heat transfer by convection. In the convection term for the equation that defines this application mode you can specify the velocity vector as an analytical expression. Alternatively, you can connect it directly to the solution of the equations of motion, for example, through a multiphysics coupling to the Incompressible Navier-Stokes application mode. The Convection and Conduction application mode is suitable for modeling of heat transfer in fluids.

## *Variables and Space Dimensions*

---

The Convection and Conduction application mode is available in 1D, 2D, 3D, Axial symmetry 1D, and Axial symmetry 2D.

---

**Note:** The optional Chemical Engineering Module also contains this Convection and Conduction application mode. In addition to the above-mentioned space dimensions, it features pseudo-2D and pseudo-3D geometry options in which it uses time as a second or third spatial dimension. This feature is useful in situations where convection in the direction of the flow is large. This often applies to reactors and equipment for unit operations.

---

## *PDE Formulation*

---

This application mode supports the following nonconservative formulation of the transient PDE for heat transfer by convection and conduction:

$$\delta_{ts} \rho C_p \frac{\partial T}{\partial t} + \nabla \cdot (-k \nabla T) = Q - \rho C_p \mathbf{u} \cdot \nabla T$$

## Subdomain Settings

---

The following table contains the quantities in the equations:

COEFFICIENT	VARIABLE	DESCRIPTION
$\delta_{ts}$	Dts_T	Time-scaling coefficient
$\rho$	rho_T	Density
$C_p$	C_T	Heat capacity
$k$	k_T	Thermal conductivity
$\underline{k}$		Thermal conductivity tensor
$k_{ij}$	kxixj_T	Thermal conductivity tensor, $x_i x_j$ component
$Q$	Q_T	Heat source
$u, v, w$	u_T, v_T, w_T	Velocity in the $x_1$ , $x_2$ , and $x_3$ directions

---

**Note:** The time-scaling coefficient,  $\delta_{ts}$ , is not available together with GLS, SUPG, or crosswind diffusion.

---

For equations in 2D or 3D, pay special attention to the isotropic thermal conductivity,  $k$ . If you select this coefficient, the application mode expands it to the diagonal of the thermal conductivity tensor, that is,  $k_{ii}$  equals  $k$ .



## STABILIZATION

Figure 7-1 shows the **Stabilization** page of the **Subdomain Settings** dialog box.

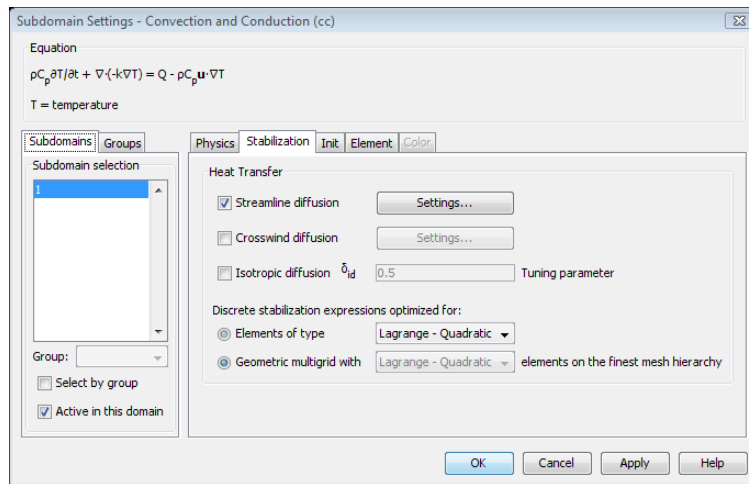


Figure 7-1: The **Stabilization** page of the **Subdomain Settings** dialog box.

The Convection and Conduction application mode supports artificial diffusion using the following methods:

- Isotropic diffusion
- Streamline diffusion
- Crosswind diffusion

Stabilization is needed because pure Galerkin discretization is unstable for convection-dominated or source-term dominated transport equations. The stabilization methods and their effects on the numerical solution are described in Chapter 17, “Stabilization Techniques.”

For optimal functionality, the exact weak formulations and constants of GLS, SUPG, and crosswind diffusion depend on the order of the basis functions (elements). You can control this dependence through two option buttons, each with its corresponding list, at the bottom of the **Stabilization** page shown in Figure 7-1. In most cases, the model uses only one element type. If this is the case, select this type from the **Elements of type** list. This is the default setting for all dimensions but 3D.

However, when using geometric multigrid (either as a solver or as a preconditioner), the equations can be discretized with different elements on the different multigrid

hierarchies. Then, select the element type that is used on the finest mesh hierarchy from the **Geometric multigrid with** list. This is the default setting for 3D, because the default solver for General Heat in 3D is BiCGStab with geometric multigrid as preconditioner. The resulting weak expressions then take forms adapted to multigrid hierarchies created in a way equivalent to the hierarchy generation method **Lower element order first (any)** (see “Constructing a Multigrid Hierarchy” on page 558 of the *COMSOL Multiphysics Reference Guide*).

The values of the GLS and SUPG constants follow Ref. 2 and Ref. 3 and cannot be changed without performance losses. In contrast, the constant  $C^e$  for Crosswind diffusion (see Equation 7-1) is a tuning constant. Ref. 4 gives the recommendations 0.7 for quadratic elements and 0.35 for linear elements. In general, the higher the value of this constant, the more diffusion it introduces. COMSOL uses the values for  $C^e$  listed in Table 7-1.

TABLE 7-1: VALUES OF  $C^e$  FOR CROSSWIND DIFFUSION IN COMSOL MULTIPHYSICS

ELEMENT ORDER	$C^e$
1	0.7
2	0.4
3	0.2
4	0.1
5	0.05

### *Streamline Diffusion*

Streamline diffusion is active by default. Figure 7-2 shows the **Streamline Diffusion Settings** dialog box. There are three types of streamline diffusion available:

- Galerkin least-squares (GLS)—GLS is the default choice and is by all references superior to any of the other methods available.

- Streamline upwind Petrov-Galerkin (SUPG)—a subset of GLS that stabilizes only with respect to convection, not reaction as well. SUPG is computationally less expensive than GLS but its stabilization properties are not as good (Ref. 1).
- Anisotropic diffusion—this is the only streamline diffusion method that needs a user-defined constant. The **Tuning parameter** edit field becomes active when you select **Anisotropic diffusion** from the list.

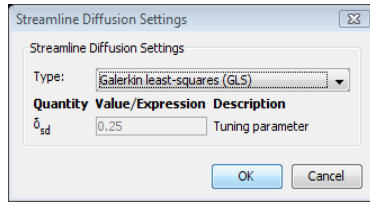


Figure 7-2: The Streamline Diffusion Settings dialog box.

The constants of GLS and SUPG are controlled as described in the section “Streamline Diffusion” on page 501. For anisotropic diffusion, there is no theory on what the value of this constant is, and the optimal value depends on the model.

#### Crosswind Diffusion

Crosswind diffusion provides extra diffusion in the region of sharp gradients. The method adds the following contribution to the weak formulation (Ref. 4):

$$-\sum_{e=1}^{N_{el}} \int_{\Omega^e} \frac{1}{2} \max\left(0, C^e - \frac{2k}{h|\beta|}\right) \frac{h|R|}{|\nabla T|} \nabla \hat{T} \left( I - \frac{\mathbf{u} \otimes \mathbf{u}}{|\mathbf{u}|^2} \right) \nabla T d\Omega \quad (7-1)$$

where  $R$  is the PDE residual,  $T$  is the test function for  $T$ , and

$$\beta = \begin{cases} \frac{\rho C_p (\mathbf{u} \cdot \nabla T)}{|\nabla T|^2} \nabla T & \text{if } \nabla T \neq 0 \\ 0 & \text{if } \nabla T = 0 \end{cases} \quad (7-2)$$

The added diffusion is orthogonal to the streamline diffusion, so you can use streamline diffusion and crosswind diffusion simultaneously. Scalar stabilization does not explicitly take any multiphysics coupling into account and is therefore the appropriate choice if the temperature equation is the only equation solved for; the temperature is a passive scalar (that is, if other equations are completely independent of the temperature); or if other equations depend only weakly on the temperature.

Figure 7-3 shows the **Crosswind Diffusion Settings** dialog box.

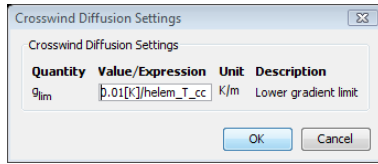


Figure 7-3: The Crosswind Diffusion Settings dialog box.

The variable  $g_{\text{lim}}$  is needed because both Equation 7-1 and Equation 7-2 contain terms of the form  $1/|\nabla T|$ , which become singular if  $\nabla T = 0$ . Hence, all occurrences of  $1/|\nabla T|$  are replaced by  $1/\max(|\nabla T|, g_{\text{lim}})$  where  $g_{\text{lim}}$  is a measure of a small gradient.

### Isotropic Diffusion

Isotropic diffusion is described in the section “Isotropic Diffusion” on page 500.

### Boundary Conditions

The available boundary conditions are:

BOUNDARY CONDITION	DESCRIPTION
$T = T_0$	Temperature
$-\mathbf{n} \cdot (-k \nabla T) = q_0$	Heat flux
$\mathbf{n} \cdot (-k \nabla T) = 0$	Insulation/symmetry
$\mathbf{n} \cdot (-k \nabla T) = 0$	Convective flux
$\mathbf{n} \cdot (-k \nabla T) = 0$	Axial symmetry
$-\mathbf{n}_1 \cdot \mathbf{q}_1 - \mathbf{n}_2 \cdot \mathbf{q}_2 = 0$	Continuity
$-\mathbf{n}_1 \cdot \mathbf{q}_1 - \mathbf{n}_2 \cdot \mathbf{q}_2 = q$	Heat flux discontinuity
$\mathbf{n}_1 \cdot (k_1 \nabla T_1) = \frac{k}{d}(T_2 - T_1)$	Thin thermally resistive layer
$\mathbf{n}_2 \cdot (k_2 \nabla T_2) = \frac{k}{d}(T_1 - T_2)$	

The Continuity and Heat flux discontinuity boundary conditions are available on interior boundaries and assembly pairs only. The Thin thermally resistive layer boundary condition is only available on assembly pairs.

In cases where convection across a boundary is much greater than diffusion, use the convective flux boundary condition. It sets the diffusive flux at the boundary to zero, but it allows convective flux to exit the domain.

---

**Note:** When working in an axisymmetry application mode, use the axial symmetry boundary condition only on the symmetry axis.

---

### *Application Mode Variables*

The Convection and Conduction application mode uses the following expressions and coefficients in boundary conditions, equations, and for postprocessing purposes.

NAME	TYPE	DESCRIPTION	EXPRESSION
T	S/B	Temperature	$T$
grad_T	S/V	Temperature gradient	$ \nabla T , \frac{\partial T}{\partial x_i}$
dflux_T	S	Conductive flux	$ \underline{k}\nabla T $
cflux_T	S	Convective flux	$ \rho C_p T \mathbf{u} $
tflux_T	S	Total heat flux	$ \underline{k}\nabla T + \rho C_p T \mathbf{u} $
ndflux_T	B	Normal conductive flux	$\mathbf{n} \cdot (-\underline{k}\nabla T)$
ncflux_T	B	Normal convective flux	$\rho C_p T \mathbf{n} \cdot \mathbf{u}$
ntflux_T	B	Normal total heat flux	$\mathbf{n} \cdot (-\underline{k}\nabla T + \rho C_p T \mathbf{u})$
dflux_T_xi	V	Conductive flux, $x_i$ component	$\sum_j -k_{ij} \frac{\partial T}{\partial x_j}$
cflux_T_xi	V	Convective flux, $x_i$ component	$\rho C_p T u_i$
tflux_T_xi	V	Total heat flux, $x_i$ component	$\sum_j -k_{ij} \frac{\partial T}{\partial x_j} + \rho C_p T u_i$
cellPe_T	S	Cell Peclet number	$\left  \frac{\rho C_p \mathbf{u} h}{\underline{k}} \right $
Dts_T	S	Time-scale factor	$\delta_{ts}$

NAME	TYPE	DESCRIPTION	EXPRESSION
rho_T	S	Density	$\rho$
C_T	S	Heat capacity	$C_p$
k_T, kx <sub>i</sub> x <sub>j</sub> _T	S	Thermal conductivity	$k, k_{ij}$
Q_T	S	Heat source	$Q$
u_T, v_T, w_T	S	Velocity of $e, x_i$ component	$u_i$
Dm_T	S	Mean diffusion coefficient	$\frac{\sum_{i,j} k_{ij} \beta_i \beta_j}{ \beta }$
res_T_cc	S	Equation residual	$\nabla \cdot (-\underline{k} \nabla T + \rho C_p T \mathbf{u}) - Q$
res_sc_T_cc	S	Shock-capturing residual	$\nabla \cdot (\rho C_p T \mathbf{u}) - Q$
da_T	S	Total time-scale factor	$\delta_{ts} \rho C_p$
q	B	Inward heat flux	$q_0$
T0	B	Prescribed temperature	$T_0$

**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: cc), for example, `tf1ux_T_cc`. (This does not apply to the dependent variable for the temperature.)

The vector variables, indicated by  $\mathbf{V}$  in the Type column, are not present in 1D versions of the Convection and Conduction application mode.

### References

1. R. P. Bonet Chaple, “Numerical Stabilization of Convection-Diffusion-Reaction Problems,” *DIAM Report, Delft University of Technology*, 2006.
2. I. Harari and T.J.R. Hughes, “What are  $C$  and  $h$ ? Inequalities for the analysis and design of finite element methods,” *Comp.Meth. Appl. Mech. Engrg*, vol. 97, pp. 157–192, 1992.
3. Y. Bazilevs, V.M. Calo, T.E. Tezduyar, and T.J.R. Hughes, “YZ $\beta$  discontinuity capturing for advection-dominated processes with application to arterial drug delivery,” *Int.J.Num. Meth. Fluids*, vol. 54, pp. 593–608, 2007.

4. R. Codina, “Comparison of some finite element methods for solving the diffusion-convection-reaction equation,” *Comp. Meth. Appl. Mech. Engrg*, vol. 156, pp. 185–210, 1998.

# Examples of Heat Transfer Models

The following heat transfer benchmark examples show how to model heat transfer using:

- Steady-state and transient analysis
- Temperature, heat flux, convective cooling, and radiation boundary conditions
- Thermal conductivity as a function of temperature

All examples are taken from a NAFEMS benchmark collection (Ref. 2).

## *1D Steady-State Heat Transfer with Radiation*

---

The first example shows a 1D steady-state thermal analysis including radiation to a prescribed ambient temperature.

### *Model Definition*

---

This 1D model has a domain of length 0.1 m. The left end is kept at 1000 K, and at the right end there is radiation to 300 K. For the radiation, the model properties are:

- The emissivity,  $\epsilon$ , is 0.98.
- The Stefan-Boltzmann constant,  $\sigma$ , is  $5.67 \cdot 10^{-8} \text{ W}/(\text{m}^2 \cdot \text{K}^4)$ .

In the domain, use the following material property:

- The thermal conductivity is 55.563 W/(m·K).



## Results

---

The following plot shows the temperature as a function of position:

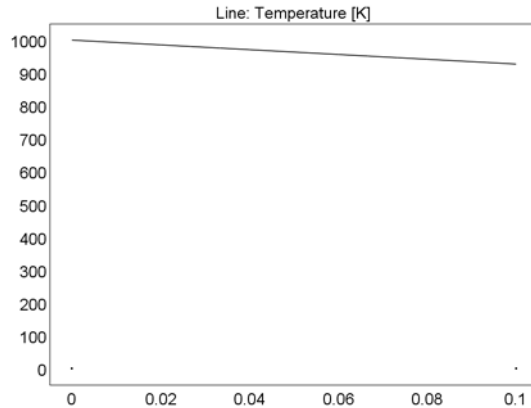


Figure 7-4: Temperature as a function of position.

The benchmark result for the right end is a temperature of 927.0 K. The COMSOL Multiphysics model, using a default mesh with 15 elements, gives a temperature at the end as 926.97 K, which is the exact benchmark value to four significant digits.

---

**Model Library path:** COMSOL\_Multiphysics/Heat\_Transfer/  
heat\_radiation\_1d

---

## Modeling Using the Graphical User Interface

---

### MODEL NAVIGATOR

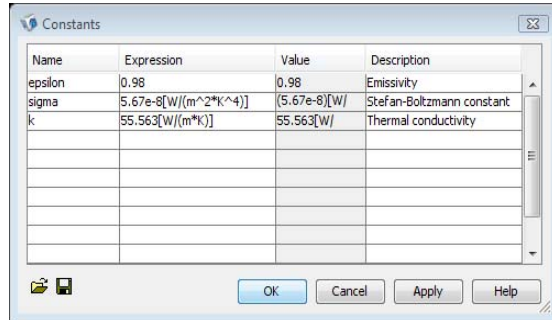
- 1 Go to the **Model Navigator** and select **ID** from the **Space dimension** list.
- 2 In the list of application modes, open the **COMSOL Multiphysics>Heat Transfer** folder and then the **Conduction** node.
- 3 Select **Steady-state analysis**.
- 4 Click **OK**.

### OPTIONS AND SETTINGS

- 1 Go to the **Options** menu and choose **Constants**.

2 Enter the following constants (the descriptions are optional); when done, click **OK**.

NAME	EXPRESSION	DESCRIPTION
epsilon	0.98	Emissivity
sigma	$5.67e-8[W/(m^2 \cdot K^4)]$	Stefan-Boltzmann constant
k	$55.563[W/(m \cdot K)]$	Thermal conductivity



## GEOMETRY MODELING

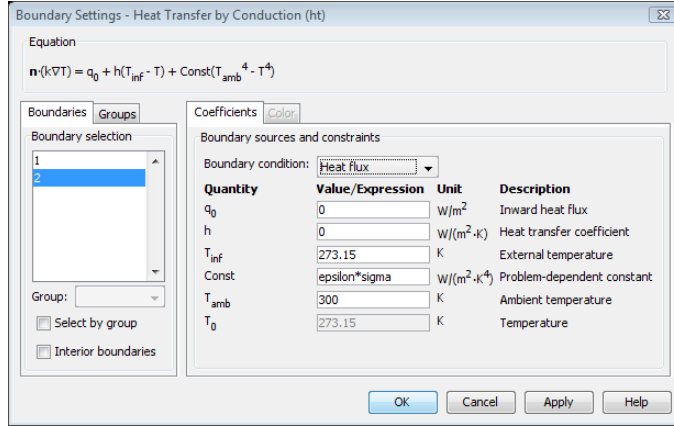
- 1 Go to the **Draw** menu, point to **Specify Objects** and click **Line**.
- 2 In the **Line** dialog box, type 0 0.1 in the **x** edit field under **Coordinates**.
- 3 Click **OK**.
- 4 Click the **Zoom Extents** button on the Main toolbar.

## PHYSICS SETTINGS

### *Boundary Conditions*

- 1 Go to the **Physics** menu and choose **Boundary Settings**.
- 2 In the **Boundary Settings** dialog box, select Boundary 1.
- 3 In the **Boundary condition** list, select **Temperature**.
- 4 In the **Temperature** edit field, type 1000.
- 5 Select Boundary 2.
- 6 From the **Boundary condition** list, select **Heat flux**.
- 7 Type  $\text{epsilon} \cdot \text{sigma}$  in the **Problem-dependent constant** edit field.
- 8 Type 300 in the **Ambient temperature** edit field.

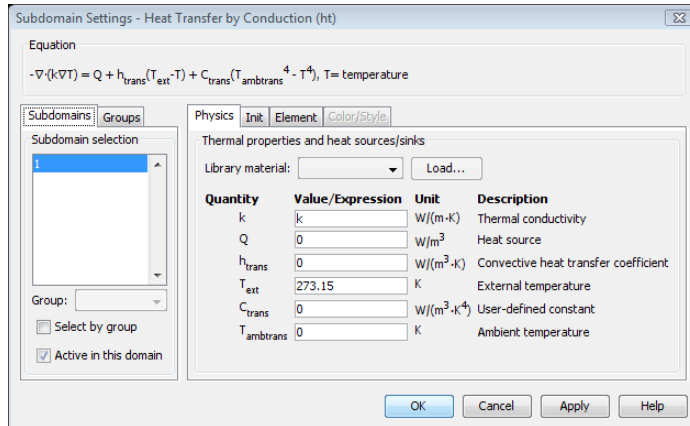
9 Click **OK**.



### Subdomain Settings

1 Go to the **Physics** menu and choose **Subdomain Settings**.

2 In the **Subdomain Settings** dialog box, enter  $k$  in the **Thermal conductivity** edit field.



Set the initial value to match the boundary condition. It serves as starting value for the nonlinear solver:

3 Click the **Init** tab.

4 Type 1000 as the initial value in the **Temperature** edit field.

5 Click **OK**.

### MESH GENERATION

Initialize the mesh by clicking the **Initialize Mesh** button on the Main toolbar.

### COMPUTING THE SOLUTION

Click the **Solve** button on the Main toolbar.

### POSTPROCESSING AND VISUALIZATION

Figure 7-4 on page 189 shows the temperature distribution in the domain. Use the zoom tools to focus on the temperature at the right end.

### *2D Steady-State Heat Transfer with Convection*

---

This example shows a 2D steady-state thermal analysis including convection to a prescribed external (ambient) temperature.

### *Model Definition*

---

This model domain is 0.6 m-by-1.0 m. For the boundary conditions:

- The left boundary is insulated.
- The lower boundary is kept at 100 °C.
- The upper and right boundaries are convecting to 0 °C with a heat transfer coefficient of 750 W/(m<sup>2</sup>·°C).

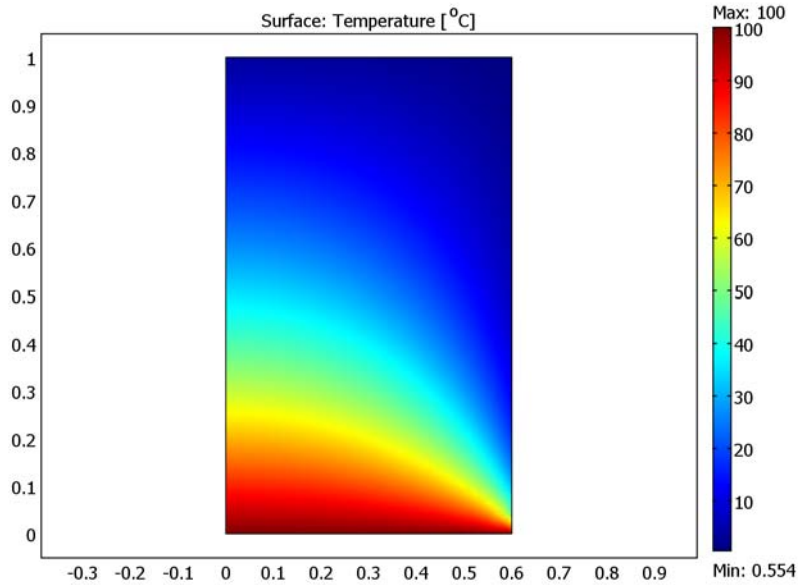
In the domain use the following material property:

- The thermal conductivity is 52 W/(m·°C).

## Results

---

The following plot shows the temperature as a function of position:



*Figure 7-5: Temperature distribution resulting from convection to a prescribed external temperature.*

The benchmark result for the target location ( $x = 0.6$  m and  $y = 0.2$  m) is a temperature of 18.25 °C. The COMSOL Multiphysics model, using a default mesh with 556 elements, gives a temperature of 18.28 °C. Successive uniform refinements show temperatures of 18.26 °C and 18.25 °C, converging toward the benchmark result.

---

**Model Library path:** COMSOL\_Multiphysics/Heat\_Transfer/  
heat\_convection\_2d

---

#### **MODEL NAVIGATOR**

- 1 Go to the **Model Navigator** and select **2D** from the **Space dimension** list.
- 2 In the list of application modes, open the **COMSOL Multiphysics>Heat Transfer** folder and then the **Conduction** node.
- 3 Select **Steady-state analysis**.
- 4 Click **OK**.

#### **GEOMETRY MODELING**

- 1 On the **Draw** menu point to **Specify Objects** and click **Rectangle**.
- 2 In the **Rectangle** dialog box, find the **Size** area and type 0.6 in the **Width** edit field, then type 1 in the **Height** edit field.
- 3 Click **OK**.
- 4 Click the **Zoom Extents** button.

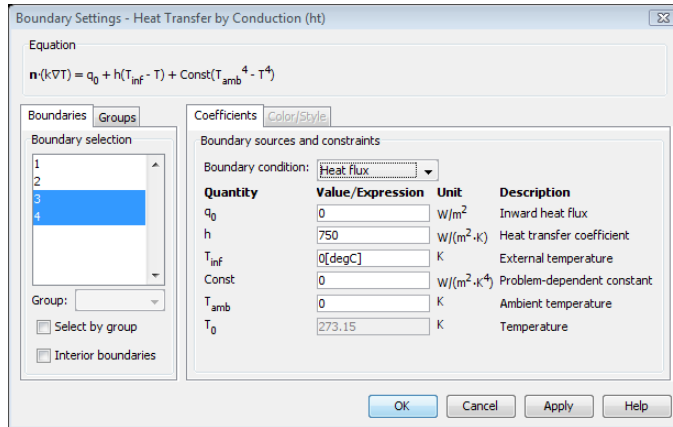
#### **PHYSICS SETTINGS**

##### *Boundary Conditions*

The default boundary condition is thermal insulation, so you must set boundary conditions for only three of the boundaries.

- 1 Go to the **Physics** menu and choose **Boundary Settings**.
- 2 In the **Boundary Settings** dialog box select Boundary 2.
- 3 In the **Boundary condition** list select **Temperature**.
- 4 Type 100[degC] in the **Temperature** edit field to specify a temperature in degrees Celsius (the default unit for temperature in the SI base unit system is kelvin).
- 5 Select Boundaries 3 and 4.
- 6 In the **Boundary condition** list select **Heat flux**.
- 7 Type 750 in the **Heat transfer coefficient** edit field.
- 8 Type 0[degC] in the **External temperature** edit field.

9 Click **OK**.



### Subdomain Settings

- 1 Go to the **Physics** menu and choose **Subdomain Settings**.
- 2 In the **Subdomain Settings** dialog box enter 52 in the **Thermal conductivity** edit field.
- 3 Click **OK**.

### MESH GENERATION

Initialize the mesh by clicking the **Initialize Mesh** button on the Main toolbar.

### COMPUTING THE SOLUTION

Click the **Solve** button on the Main toolbar.

### POSTPROCESSING AND VISUALIZATION

Figure 7-5 on page 193 shows the temperature distribution in the domain. To get a numerical value of the temperature at the reference point, do the following steps:

- 1 Go to the **Postprocessing** menu and choose **Data Display>Subdomain** to open the **Data Display** dialog box.
- 2 Select **°C** from the **Unit** list.
- 3 In the **Coordinates** area enter 0.6 in the **x** edit field and 0.2 in the **y** edit field.
- 4 Click **OK**. The temperature (about 18.28 °C) appears in the message log at the bottom of the main COMSOL Multiphysics window.

## *2D Axisymmetric Transient Heat Transfer*

---

This example shows an axisymmetric transient thermal analysis with a step change to 1000 °C at time 0.

### *Model Definition*

---

This model domain is 0.3 m-by-0.4 m. For the boundary conditions, assume the following:

- The left boundary is the symmetry axis.
- The other boundaries have a temperature of 1000 °C. The entire domain is at 0 °C at the start, which represents a step change in temperature at the boundaries.

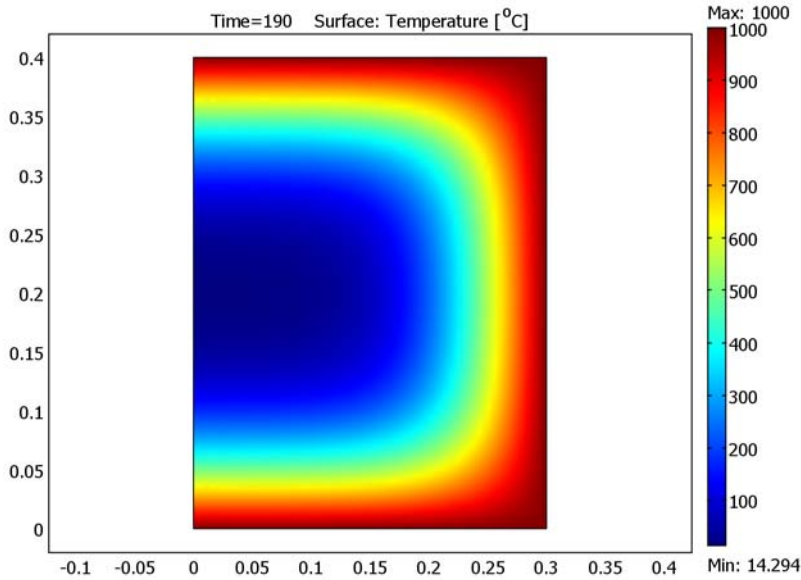
In the domain use the following material properties:

- The density,  $\rho$ , is 7850 kg/m<sup>3</sup>
- The heat capacity is 460 J/(kg·°C)
- The thermal conductivity is 52 W/(m·°C)



## Results

The following plot shows the temperature as a function of position after 190 seconds:



*Figure 7-6: Temperature distribution after 190 seconds.*

The benchmark result for the target location ( $r = 0.1$  m and  $z = 0.3$  m) is a temperature of  $186.5$  °C. The COMSOL Multiphysics model, using a default mesh with about 720 elements, gives a temperature of roughly  $186.4$  °C.

As an additional postprocessing step, map the axisymmetric solution to 3D using an extrusion coupling variable to show the solution for the entire cylinder (see Figure 7-7).

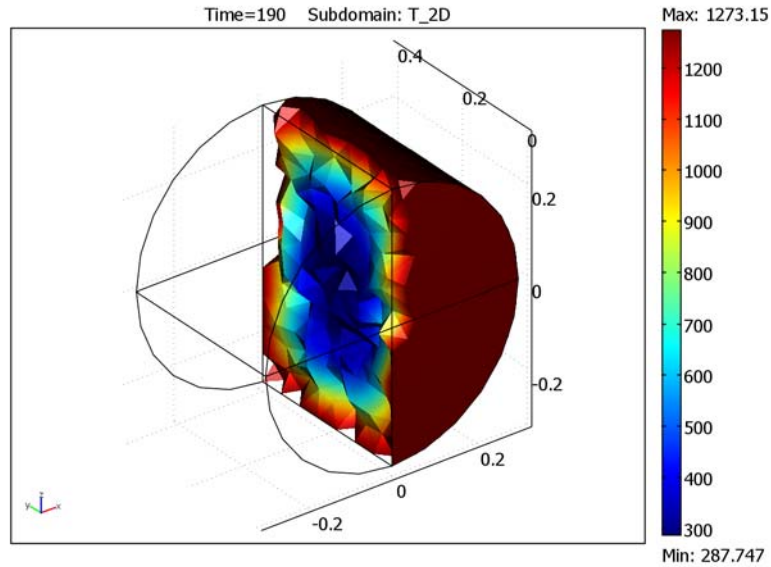


Figure 7-7: Postprocessing of the temperature in the full 3D geometry.

---

**Model Library path:** COMSOL\_Multiphysics/Heat\_Transfer/  
heat\_transient\_axl

---

### *Modeling Using the Graphical User Interface*

---

#### **MODEL NAVIGATOR**

- 1 Go to the **Model Navigator** and select **Axial symmetry (2D)** from the **Space dimension** list.
- 2 From the list of application modes, select **COMSOL Multiphysics>Heat Transfer>Conduction**.
- 3 Select **Transient analysis**.
- 4 Click **OK**.

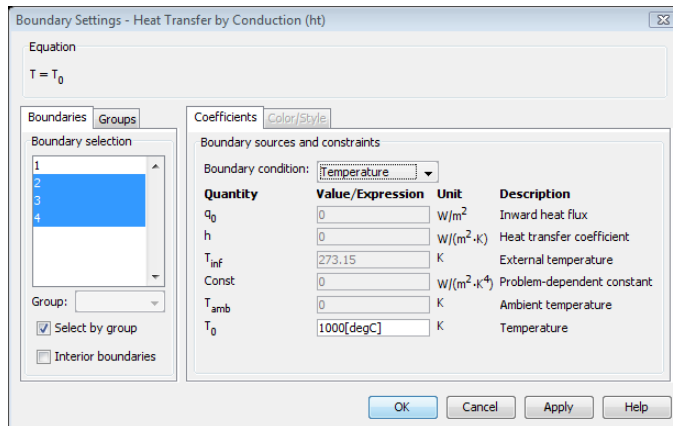
## GEOMETRY MODELING

- 1 Go to the **Draw** menu, point to **Specify Objects** and click **Rectangle**.
- 2 In the **Rectangle** dialog box go to the **Size** area and enter 0.3 in the **Width** edit field and 0.4 in the **Height** edit field.
- 3 Click **OK**.
- 4 Click the **Zoom Extents** button on the Main toolbar.

## PHYSICS SETTINGS

### *Boundary Conditions*

- 1 Go to the **Physics** menu and choose **Boundary Settings**.
- 2 In the **Boundary Settings** dialog box select Boundary 1.
- 3 In the **Boundary condition** list select **Axial symmetry**.
- 4 Select the **Select by group** check box and choose Boundaries 2, 3, and 4 by selecting one of them.
- 5 Select **Temperature** in the **Boundary condition** list.
- 6 Type 1000[degC] in the **Temperature** edit field.
- 7 Click **OK**.



### *Subdomain Settings*

- 1 Go to the **Physics** menu and choose **Subdomain Settings**.

- 2 In the **Subdomain Settings** dialog box enter the thermal properties in the domain according to the following table:

PROPERTIES	SUBDOMAIN I
k (isotropic)	52
$\rho$	7850
$C_p$	460

- 3 Click the **Init** tab.
- 4 Type 0[degC] in the **T(t<sub>0</sub>)** edit field for the initial temperature.
- 5 Click **OK**.

### MESH GENERATION

Initialize the mesh by clicking the **Initialize Mesh** button on the Main toolbar.

### COMPUTING THE SOLUTION

- 1 Go to the **Solve** menu and choose **Solver Parameters**.
- 2 In the **Time stepping** area in the **Solver Parameters** dialog box enter 0:10:190 in the **Times** edit field.
- 3 Click **OK**.
- 4 Click the **Solve** button on the Main toolbar.

### POSTPROCESSING AND VISUALIZATION

#### *Getting the Reference Temperature*

Figure 7-6 on page 197 shows the temperature distribution in the domain. To get the numerical value at the reference point, use the **Data Display** dialog box:

- 1 Choose **Postprocessing>Data Display>Subdomain**.
- 2 In the **Coordinates** area enter 0.1 in the **r** edit field and 0.3 in the **z** edit field.
- 3 Click **OK**.

The value of the temperature in the reference point appears in the message log. You can also click in the temperature plot at (0.1, 0.3) to display the temperature at that point in the message log.

### *Postprocessing in 3D*

To postprocess the solution in 3D, first revolve the geometry into a cylinder in a 3D geometry and then map the axisymmetric solution to the cylinder using an extrusion coupling variable:

- 1 From the **Draw** menu, choose **Revolve**.
- 2 In the **Revolve** dialog box, leave the default settings and click **OK**. This creates a cylinder in 3D. Note that the axis of revolution in 3D is the  $y$ -axis, which means that the plane that you map the radial coordinate  $r$  to is the  $xz$ -plane.
- 3 Click the **Geom1** tab at the top of the drawing area to return to the 2D axisymmetric geometry.
- 4 Choose **Options>Extrusion Coupling Variables>Subdomain Variables**.
- 5 In the **Subdomain Extrusion Variables** dialog box, select Subdomain 1 and then type  $T_{2D}$  in the first row of the **Name** column and  $T$  is the first row of the **Expression** column. This creates an extrusion coupling variable  $T_{2D}$  that represents the temperature (the variable  $T$ ).  
  
Click the **General transformation** button. The default source transformation ( $\mathbf{x}$ :  $\mathbf{r}$  and  $\mathbf{y}$ :  $\mathbf{z}$ ) is correct.
- 6 Click the **Destination** tab.
- 7 Select **Geom2** from the **Geometry** list, select **Subdomain** from the **Level** list, and finally select the **I** check box for Subdomain 1 in the Subdomain selection list. The variable  $T_{2D}$  is the only extrusion coupling variable and the software selects it automatically.
- 8 In the **Destination transformation** area, type  $\sqrt{x^2+z^2}$  in the  $\mathbf{x}$  edit field, and leave the value  $\frac{y}{\sqrt{x^2+z^2}}$  in the  $\mathbf{y}$  edit field. This transforms  $r$  and  $z$  in the axisymmetric geometry to  $\sqrt{x^2+z^2}$  and  $y$ , respectively, in the 3D geometry.
- 9 Click **OK**.
- 10 From the **Solve** menu, choose **Update Model** to map the solution to the 3D geometry.
- 11 From the **Postprocessing** menu, choose **Plot Parameters**.
- 12 To plot the temperature in the cylinder as a subdomain plot, clear the **Slice** check box and select the **Subdomain** check box in the **Plot type** area on the **General** page.
- 13 Select the **Element selection** check box and type  $x>0$  in the **Logical expression for inclusion** edit field to “look inside” by removing all elements with a negative  $x$  coordinate.
- 14 Click the **Subdomain** tab, and then type  $T_{2D}$  in the **Expression** edit field.

**15** Click **OK** to generate Figure 7-7 on page 198. To reproduce that plot, also click the **Headlight** button on the Camera toolbar.

### *References*

---

1. Frank P. Incropera and David P. DeWitt, *Fundamentals of Heat and Mass Transfer*, 4th edition, John Wiley & Sons, New York, 1996.
2. A.D. Cameron, J. A. Casey, and G.B. Simpson: *NAFEMS Benchmark Tests for Thermal Analysis (Summary)*, NAFEMS Ltd., Glasgow, 1986.

# Structural Mechanics

This chapter explains how to use the Structural Mechanics application modes to simulate and analyze applications involving solid mechanics. It begins with a brief theoretical backgrounder on structural mechanics, after which subsequent sections give details of the application modes.

# The Structural Mechanics Application Modes

COMSOL Multiphysics includes four application modes for stress analysis and general structural mechanics simulation:

- The Solid, Stress-Strain application mode (for 3D geometries)
- The Plane Stress application mode (for 2D geometries)
- The Plane Strain application mode (for 2D geometries)
- The Axial Symmetry Stress-Strain application mode (for 2D axisymmetric geometries)

The last three cases are 2D simplifications of the full 3D equations, simplifications that are valid under certain assumptions.

---

**Note:** The optional Structural Mechanics Module contains application modes and models that allow for extended, specialized analyses of structural and solid mechanics problems.

---



# Theory Background

## *Strain-Displacement Relationship*

---

It is possible to completely describe the strain conditions at a point with the deformation components— $(u, v, w)$  in 3D—and their derivatives. You can express the shear strain in a tensor form,  $\epsilon_{xy}$ ,  $\epsilon_{yz}$ ,  $\epsilon_{xz}$ , or in an engineering form,  $\gamma_{xy}$ ,  $\gamma_{yz}$ ,  $\gamma_{xz}$ . Following the small-displacement assumption, the normal strain components and the shear strain components are given from the deformation as follows:

$$\begin{aligned}\epsilon_x &= \frac{\partial u}{\partial x} & \epsilon_{xy} &= \frac{\gamma_{xy}}{2} = \frac{1}{2} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\ \epsilon_y &= \frac{\partial v}{\partial y} & \epsilon_{yz} &= \frac{\gamma_{yz}}{2} = \frac{1}{2} \left( \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y} \right) \\ \epsilon_z &= \frac{\partial w}{\partial z} & \epsilon_{xz} &= \frac{\gamma_{xz}}{2} = \frac{1}{2} \left( \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x} \right)\end{aligned}$$

The symmetric strain tensor  $\epsilon$  consists of both normal and shear strain components:

$$\epsilon = \begin{bmatrix} \epsilon_x & \epsilon_{xy} & \epsilon_{xz} \\ \epsilon_{xy} & \epsilon_y & \epsilon_{yz} \\ \epsilon_{xz} & \epsilon_{yz} & \epsilon_z \end{bmatrix}$$

## *Stress-Strain Relationship*

---

The stress in a material is described by the symmetric stress tensor

$$\sigma = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z \end{bmatrix} \quad \tau_{xy} = \tau_{yx} \quad \tau_{xz} = \tau_{zx} \quad \tau_{yz} = \tau_{zy}$$

consisting of three normal stresses ( $\sigma_x$ ,  $\sigma_y$ ,  $\sigma_z$ ) and six, or if symmetry is used, three shear stresses ( $\tau_{xy}$ ,  $\tau_{yz}$ ,  $\tau_{xz}$ ). The stress-strain relationship for linear conditions reads:

$$\sigma = D\epsilon$$

where  $D$  is the  $6 \times 6$  elasticity matrix, and the stress and strain components are described in vector form with the six stress and strain components in column vectors defined as

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \\ \tau_{xy} \\ \tau_{yz} \\ \tau_{xz} \end{bmatrix} \quad \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_x \\ \varepsilon_y \\ \varepsilon_z \\ \gamma_{xy} \\ \gamma_{yz} \\ \gamma_{xz} \end{bmatrix}$$

---

**Note:** The following descriptions use the compact notation  $\boldsymbol{\sigma}$  and  $\boldsymbol{\varepsilon}$ , meaning either the stress/strain vector or tensor depending on the context.

---

The elasticity matrix  $D$  and the more basic matrix  $D^{-1}$  (the inverse of  $D$ , also known as the flexibility or compliance matrix) are defined differently for isotropic, orthotropic, and anisotropic material. For isotropic materials, the  $D^{-1}$  matrix looks like

$$D^{-1} = \frac{1}{E} \begin{bmatrix} 1 & -\nu & -\nu & 0 & 0 & 0 \\ -\nu & 1 & -\nu & 0 & 0 & 0 \\ -\nu & -\nu & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2(1+\nu) & 0 & 0 \\ 0 & 0 & 0 & 0 & 2(1+\nu) & 0 \\ 0 & 0 & 0 & 0 & 0 & 2(1+\nu) \end{bmatrix}$$

where  $E$  is the modulus of elasticity (also known as *Young's modulus*), and  $\nu$  is *Poisson's ratio*, which defines contraction in the perpendicular direction. Inverting  $D^{-1}$

$$D = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix}$$

### *Implementation*

---

COMSOL Multiphysics bases its implementation of the structural mechanics application modes on a weak formulation of the equilibrium equations expressed in the global stress components.

#### **EQUILIBRIUM EQUATION**

The equilibrium equations expressed in the stresses for 3D are

$$\begin{aligned} -\frac{\partial \sigma_x}{\partial x} - \frac{\partial \tau_{xy}}{\partial y} - \frac{\partial \tau_{xz}}{\partial z} &= F_x \\ -\frac{\partial \tau_{xy}}{\partial x} - \frac{\partial \sigma_y}{\partial y} - \frac{\partial \tau_{yz}}{\partial z} &= F_y \\ -\frac{\partial \tau_{xz}}{\partial x} - \frac{\partial \tau_{yz}}{\partial y} - \frac{\partial \sigma_z}{\partial z} &= F_z \end{aligned}$$

where  $\mathbf{F}$  denotes the volume forces (body forces).

Using compact notation, you can write this relationship as

$$-\nabla \cdot \boldsymbol{\sigma} = \mathbf{F}$$

where  $\boldsymbol{\sigma}$  is the stress tensor. Substituting the stress-strain and strain-displacement relationships in the above equation results in Navier's equation expressed in the displacement.

#### **SETTING UP EQUATIONS FOR THE DIFFERENT ANALYSES**

All the application modes for structural mechanics support the following analysis types:

- Static analysis

- Eigenfrequency analysis
- Transient analysis

The equations to solve and the solvers to use vary depending on the analysis type. You specify them by selecting an analysis type in the **Model Navigator** or during modeling in the **Application Mode Properties** dialog box or the **Solver Parameters** dialog box.

#### *Static Analysis*

These analysis types all use the same equation but invoke a different solver. The following discussion uses static analyses to represent all these types of analysis because they use the same equations.

Substitution of the stress-strain relationship and the strain-displacement relationship into the static equilibrium equation produces Navier's equation of equilibrium expressed in the displacements.

For static conditions including temperature, Navier's equation reads

$$-\nabla \cdot (c \nabla \mathbf{u}) = \mathbf{F}$$

Details about the corresponding coefficients appear in the application mode implementation sections.

#### *Eigenfrequency Analysis*

The difference between this analysis type and the static analysis is that it adds the mass.

If you want to solve for the eigenfrequencies, use the following formulation:

$$-\lambda d_a \mathbf{u} - \nabla \cdot c_{sta} \nabla \mathbf{u} = \mathbf{0}$$

The eigenfrequency  $f$  relates to the eigenvalue  $\lambda$  as

$$f = \frac{\sqrt{\lambda}}{2\pi}$$

The density appears in the  $d_a$  coefficient. In COMSOL Multiphysics you can work with either eigenvalues or eigenfrequencies.

#### *Transient Analysis*

A transient problem requires the introduction of Newton's Second Law.

$$\rho \frac{\partial^2 \mathbf{u}}{\partial t^2} - \nabla \cdot c \nabla \mathbf{u} = \mathbf{F}$$

COMSOL Multiphysics provides Rayleigh damping to model viscous damping by specifying two damping coefficients. The default is to use no damping. To activate Rayleigh damping, see “Damping” on page 215.

To see how the software handles the Rayleigh damping, consider a system with a single degree of freedom. You can describe the motion for a system with viscous damping and with a single degree of freedom as

$$m \frac{d^2 u}{dt^2} + \xi \frac{du}{dt} + ku = f(t) \quad (8-1)$$

The Rayleigh damping model expresses the damping parameter  $\xi$  in terms of the mass  $m$  and the stiffness  $k$  as

$$\xi = \alpha_{dM} m + \beta_{dK} k$$

You can reduce the 2nd-order system in Equation 8-1 to a 1st-order system in time by introducing new variables  $\mathbf{v} = (u_t, v_t, w_t)^t$ :

$$\mathbf{v} = \frac{\partial}{\partial t} \mathbf{u}$$

The expanded system look like the following, with the Rayleigh damping modeled in an equivalent way to the single-degree-of-freedom system shown above:

$$\begin{bmatrix} I & 0 \\ 0 & \rho \end{bmatrix} \frac{\partial}{\partial t} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} - \nabla \cdot \left[ \begin{bmatrix} 0 & 0 \\ c_{sta} & c_{sta} \beta_{dK} \end{bmatrix} \nabla \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} \right] + \begin{bmatrix} 0 & -I \\ 0 & \alpha_{dM} \rho \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{K} \end{bmatrix}$$

The difference between this system and the scalar case with just one degree of freedom is that the damping parameters are defined locally, meaning that they can vary in space and between subdomains.

# Application Mode Descriptions

This section describes how to define a structural mechanics model using the application modes. It consists of the following sections:

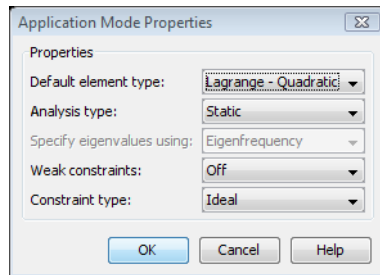
- Application mode properties
- Material properties
- Constraints
- Loads

The dialog boxes and options are similar for all of the application modes. However, the number of independent variables (spatial coordinates) and dependent variables (displacements) as well as their names vary depending on the application mode. For details on additional application-specific properties, see the following sections, which describe the application modes.

## *Application Mode Properties*

---

To specify application mode properties, choose **Properties** from the **Physics** menu to open the **Application Mode Properties** dialog box.



In this dialog box you can specify different global settings for the model:

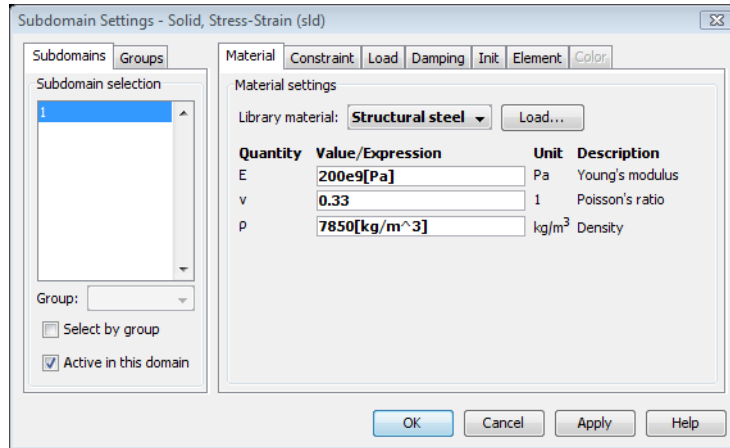
- Make a selection from the **Default element type** list. This selection becomes the default on all subdomains. Available elements are:
  - **Lagrange - Linear**, 1st-order Lagrange elements
  - **Lagrange - Quadratic**, 2nd-order Lagrange elements
  - **Lagrange - Cubic**, 3rd-order Lagrange elements
  - **Lagrange - Quartic**, 4th-order Lagrange elements
  - **Lagrange - Quintic**, 5th-order Lagrange elements
- Make a selection from the **Analysis type** list; the default is static analysis. Your choice affects the equations and which solver COMSOL Multiphysics uses if you previously selected the **Auto select solver** check box in the **Solver Parameters** dialog box.
- Indicate how you prefer to specify eigenvalues—using eigenvalues themselves or eigenfrequencies—by selecting from the **Specify eigenvalues using** list. This property is applicable only to eigenfrequency analyses.
- Control whether or not weak constraints should be available in the **Weak constraints** list by selecting **On** or **Off**.

Use weak constraints for accurate computation of reaction forces, solving for the reaction force, which is equal to the Lagrange multiplier that the weak constraint introduces as an additional dependent variable. When weak constraints are enabled, all constraints are weak constraints by default, but it is possible to change this setting for individual domains.

- Choose the type of constraint. Select **Ideal** or **Non-ideal** from the **Constraint type** list. See “Ideal vs. Non-Ideal Constraints” on page 359 for more information about these constraint types.

## Material Properties

You define material properties in a model by going to the **Subdomain Settings** dialog box and then the **Material** page.



The material properties for the structural mechanics application modes appear in the table below.

PARAMETER	VARIABLE	DESCRIPTION	COMMENT
$E$	E	Young's modulus	
$\nu$	nu	Poisson's ratio	
$\rho$	rho	Density	
thickness	thickness	Thickness of the structure	Plane stress and plane strain only

**Young's modulus** Defines the material's modulus of elasticity,  $E$ . For isotropic materials it is the spring stiffness in Hooke's law, shown in 1D form as

$$\sigma = E\varepsilon$$

where  $\sigma$  is the stress and  $\varepsilon$  is the strain.

**Poisson's ratio** Denoted by  $\nu$ , it defines the normal strain in the perpendicular direction generated from a normal strain in the other direction.

$$\varepsilon_{\perp} = -\nu\varepsilon_{\parallel}$$



**Density** This property,  $\rho$ , specifies the material's density.

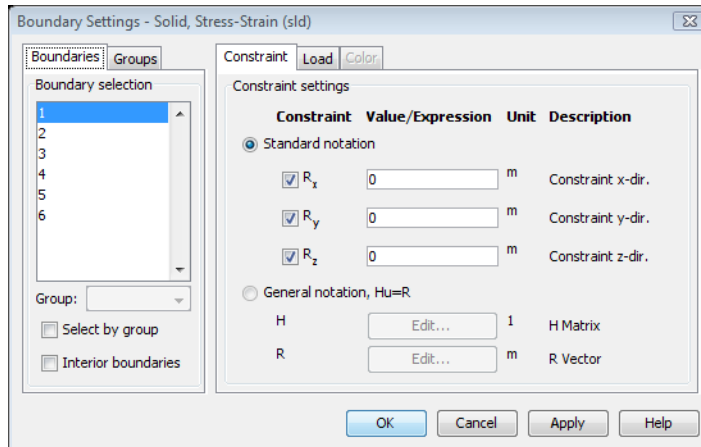
The COMSOL Multiphysics materials library contains values for many of these properties for common materials such as steel, copper, or aluminum. See “Using the Materials/Coefficients Library” on page 229 in the *COMSOL Multiphysics User's Guide*.

## Constraints

A constraint specifies the displacement of certain parts of a structure. You can define constraints on all domain levels:

- Points, edges, boundaries (faces), and subdomains in 3D
- Points, boundaries, and subdomains in 2D.

To specify them, go to the **Constraint** page in the **Subdomain Settings**, **Boundary Settings**, **Edge Settings**, or **Point Settings** dialog boxes. The following image shows the **Boundary Settings** dialog box for the Solid, Stress-Strain application mode, but this page looks the same on all domain levels in all structural mechanics application modes.



You can describe a constraint using standard or general notation, an option you control using the **Standard notation** and the **General notation,  $H=U=R$**  buttons.

### STANDARD NOTATION FOR CONSTRAINTS

Using standard notation, you constrain each displacement direction independently:

- 1 Select the check boxes in front of  **$R_x$** ,  **$R_y$** , and  **$R_z$**  to activate the constraint.

- 2 Enter the value or expression for the displacement in the edit fields. The default value is 0.

### GENERAL NOTATION FOR CONSTRAINTS

Using general notation (illustrated here for the 2D case), you define the  $H$  matrix and  $R$  vector in the relationship

$$H \begin{bmatrix} u \\ v \end{bmatrix} = R$$

to specify constraints as any linear combination of displacements components.

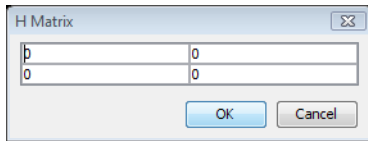
Enter the  $H$  matrix and  $R$  vector in special matrix dialog boxes by clicking the **Edit** buttons.

For example, to set the condition  $u = v$ , use the settings

$$H = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix}, \quad R = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

which allow the domain to move only diagonally in the  $x$ - $y$  plane.

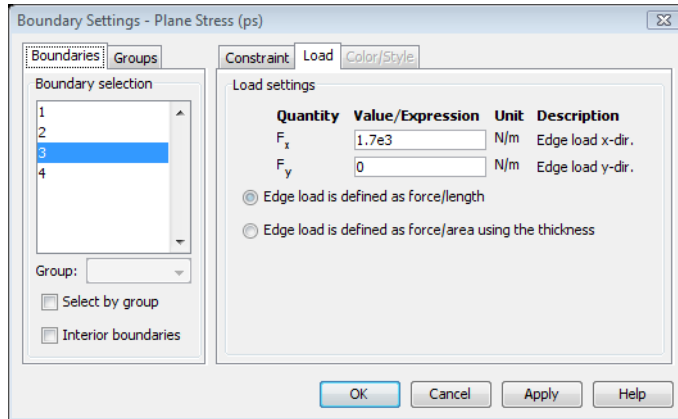
The **H Matrix** dialog box for the above example is



### *Loads*

---

A load is a general name for forces applied to a structure. You can specify loads on all domain types. To specify them, go to the **Load** page in the **Subdomain Settings**, **Boundary Settings**, **Edge Settings**, or **Point Settings** dialog boxes. The image below shows the **Boundary Settings** dialog box for the Plane Stress application mode, but this page looks similar on all domain levels in all structural mechanics application modes.



When dealing with plane stress and plane strain, you can optionally specify a load in different ways using the thickness. The following table illustrates how to define loads on different domains in different application modes; the SI unit appears in parentheses.

APPLICATION MODE	POINT	EDGE	BOUNDARY	SUBDOMAIN
Plane Stress, Plane Strain	force (N)		force/area (N/m <sup>2</sup> ) or force/length (N/m)	force/volume (N/m <sup>3</sup> ) or force/area (N/m <sup>2</sup> )
Axisymmetry, Stress-Strain	total force along the circumference (N)		force/area (N/m <sup>2</sup> )	force/volume (N/m <sup>3</sup> )
Solid, Stress-Strain	force (N)	force/length (N/m)	force/area (N/m <sup>2</sup> )	force/volume (N/m <sup>3</sup> )

## Damping

To specify damping, click the **Damping** tab in the **Subdomain Settings** dialog box. You can choose between *Rayleigh damping* and no damping (the default) in the **Damping model** list. For specifying the Rayleigh damping, there are two damping parameters:

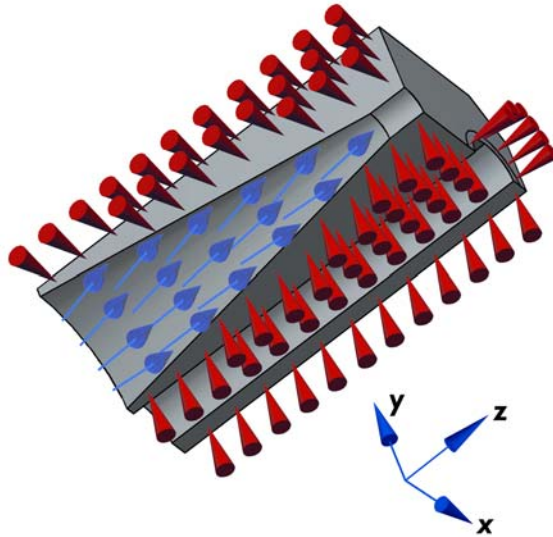
**Mass damping parameter** Defines the Rayleigh damping model's mass damping,  $\alpha_{dM}$ .

**Stiffness damping parameter** Defines the Rayleigh damping model's stiffness damping,  $\beta_{dK}$ .

$\alpha_{dM}$	alphadM	Mass damping parameter
$\beta_{dK}$	betadK	Stiffness damping parameter

# The Solid, Stress-Strain Application Mode

This section explains how to use the Solid, Stress-Strain application mode for stress and strain analysis of 3D solids.



*A 3D solid with applied loads and constraints in a stress analysis.*

## *Variables and Space Dimensions*

---

The degrees of freedom (dependent variables) are the global displacements  $u$ ,  $v$ , and  $w$  in the global  $x$ ,  $y$ , and  $z$  directions.

## *PDE Formulation*

---

COMSOL Multiphysics formulates this application mode using the equilibrium equations described in general terms in the section “Implementation” on page 207.

## APPLICATION MODE PARAMETERS

The parameters that define the loads, material, and constraints are reviewed in the section “Application Mode Descriptions” on page 210.

### *Application Mode Variables*

A large number of variables, listed below, are available for use in expressions and for postprocessing. The table uses an index convention, where a single index on  $u_i$  ( $u_i$ ) means that  $i$  runs over the global displacements ( $u, v, w$ ), whereas a single index on other names like  $s_i$  ( $\sigma_i$ ) means that  $i$  runs over the global space variables ( $x, y, z$ ), and a double index  $s_{ij}$  ( $\tau_{ij}$ ) means that  $ij$  runs over the combinations ( $xy, yz, xz$ ) of the space variables.

In addition to the variables in the table, almost all application mode parameters are available as variables. Some are different for some analyses; see the Analysis column, which uses the following abbreviations:

ANALYSIS	ABBREVIATION
Static	S
Transient	T

TABLE 8-1: SOLID, STRESS-STRAIN APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
$u_i$	$u_i$	All	All	$x_i$ displacement	$u_i$
$u_{i\_t}$	$u_{it}$	T	All	$x_i$ velocity	$u_{it}$
disp	disp	All	All	Total displacement	$\sqrt{\sum_i (\text{real}(u_i))^2}$
ex	$\epsilon_x$	All	S	$\epsilon_x$ normal strain global system	$\frac{\partial u_i}{\partial x_i}$
$e_{ij}$	$\epsilon_{ij}$	All	S	$\epsilon_{ij}$ shear strain global coord. system	$\frac{1}{2} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$
$e_{i\_t}$	$\epsilon_{it}$	T	S	$\epsilon_{it}$ normal velocity strain global system	$\frac{\partial u_{it}}{\partial x_i}$

TABLE 8-1: SOLID, STRESS-STRAIN APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
$e_{ij\_t}$	$\epsilon_{ijt}$	T	S	$\epsilon_{ijt}$ shear velocity strain global coord. system	$\frac{1}{2} \left( \frac{\partial u_{it}}{\partial x_j} + \frac{\partial u_{jt}}{\partial x_i} \right)$
en	$\epsilon_n$	All	S	$n$ th principal strain, $n = 1, 2, 3$	
$en_i$	$\epsilon_{ni}$	All	S	$n$ th principal strain component in the $x_i$ direction, $n = 1, 2, 3$	
$s_i$	$\sigma_i$	All	S	$\sigma_i$ normal stress global coord. system	$D\epsilon$
$s_{ij}$	$\tau_{ij}$	All	S	$\tau_{ij}$ shear stress global coord. system	$D\epsilon$
$s_{i\_t}$	$\sigma_{it}$	T	S	$\sigma_{it}$ time derivative of normal stress global coord. system	$D\epsilon_t$
$s_{ij\_t}$	$\tau_{ijt}$	T	S	$\tau_{ijt}$ time derivative of shear stress global coord. system	$D\epsilon_t$
sn	$\sigma_n$	All	S	$n$ th principal stress, $n = 1, 2, 3$	
$sn_i$	$\sigma_{ni}$	All	S	$n$ th principal stress component in the $x_i$ direction, $n = 1, 2, 3$	
tresca	$\sigma_{tresca}$	All	S	Tresca stress	$\max(\max( \sigma_1 - \sigma_2 ,  \sigma_2 - \sigma_3 ),  \sigma_1 - \sigma_3 )$

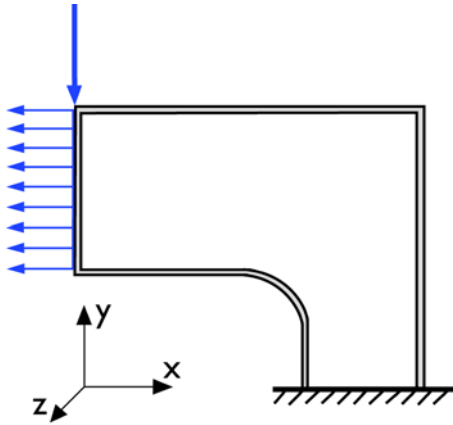
TABLE 8-1: SOLID, STRESS-STRAIN APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
mises	$\sigma_{mises}$	All	S	von Mises stress	
Ta <sub>i</sub>	Ta <sub>i</sub>	All	B	Surface traction (force/area) in x <sub>i</sub> direction	$\begin{bmatrix} T a_x \\ T a_y \\ T a_z \end{bmatrix} = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{xy} & \sigma_y & \tau_{yz} \\ \tau_{xz} & \tau_{yz} & \sigma_z \end{bmatrix} \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix}$
F <sub>ig</sub>	F <sub>ig</sub>	All	All	Body load, face load, edge load, point load, in global x <sub>i</sub> direction	$\begin{bmatrix} F_{xg} \\ F_{yg} \\ F_{zg} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}$

**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: s1d), for example, `mises_s1d`. (This does not apply to the dependent variables for the displacements.)

# The Plane Stress Application Mode

Use the Structural Mechanics Plane Stress application mode to analyze thin in-plane loaded plates. This application mode solves for the global displacements ( $u, v$ ) in the  $x$  and  $y$  directions. In a state of plane stress the  $\sigma_z$ ,  $\tau_{yz}$ , and  $\tau_{xz}$  components of the stress tensor are assumed to be zero.



*Loads and constraints in a plane stress analysis.*

Loads in the  $x$  and  $y$  directions are allowed. The mode assumes that the loads are constant throughout the thickness of the material but that thickness can vary in the  $x$  and  $y$  directions. The plane stress condition prevails in a thin flat plate in the  $xy$ -plane loaded only in its own plane and without any  $z$ -direction restraint.

## *Material*

---

An additional material parameter for plane stress is the plate's thickness.

PARAMETER	VARIABLE	DESCRIPTION	MATERIAL MODEL
th	thickness	The plate's thickness	All

## *PDE Formulation*

---

COMSOL Multiphysics formulates the application mode using the equilibrium equations described in general terms in the section “Implementation” on page 207.



## APPLICATION MODE PARAMETERS

The application mode parameters defining the loads, material and constraints are explained in the section “Application Mode Descriptions” on page 210.

### *Application Mode Variables*

A large number of variables are available for use in expressions and for postprocessing. Table 8-2 uses an index convention where a single index on  $u_i$  ( $u_i$ ) means that  $i$  runs over the global displacements ( $u, v$ ), whereas a single index on other names like  $s_i$  ( $\sigma_i$ ) means that  $i$  runs over the global space coordinates ( $x, y$ ). A single index on  $x_i$  means that  $i$  runs over the global space coordinates ( $x, y$ ).

In addition to the variables listed below, almost all application mode parameters are available as variables. Some vary for different analyses, as seen in the Analysis column.

ANALYSIS	ABBREVIATION
Static	S
Transient	T

TABLE 8-2: PLANE STRESS APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
$u_i$	$u_i$	All	All	$x_i$ displacement	$u_i$
$u_{i\_t}$	$u_{it}$	T	All	$x_i$ velocity	$u_{it}$
disp	disp	All	All	Total displacement	$\sqrt{\sum_i (\text{real}(u_i))^2}$
$e_i$	$\epsilon_i$	All	S	$\epsilon_i$ normal strain global system	$\frac{\partial u_i}{\partial x_i}$
ez	$\epsilon_z$	All	S	$\epsilon_z$ normal strain out of the $xy$ -plane	$\frac{\left( \sum_{k=1,2,4} D_{3k}(\epsilon_k) \right)}{D_{33}}$
exy	$\epsilon_{xy}$	All	S	$\epsilon_{xy}$ shear strain global coord. system	$\frac{1}{2} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)$
$e_{i\_t}$	$\epsilon_{it}$	T	S	$\epsilon_{it}$ velocity normal strain global coord. system	$\frac{\partial u_{it}}{\partial x_i}$

TABLE 8-2: PLANE STRESS APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
ez_t	$\epsilon_{zt}$	T	S	$\epsilon_{zt}$ velocity normal strain out of the $x$ - $y$ plane	$\frac{\sum_{k=1,2,4} D_{3k} \epsilon_{kt}}{D_{33}}$
exy_t	$\epsilon_{xyt}$	All	S	$\epsilon_{xyt}$ velocity shear strain global coord. system	$\frac{1}{2} \left( \frac{\partial u_t}{\partial y} + \frac{\partial v_t}{\partial x} \right)$
en	$\epsilon_n$	All	S	$n$ th principal strain, $n = 1, 2, 3$	
eni	$\epsilon_{ni}$	All	S	$n$ th principal strain component in the $x_i$ direction, $n = 1, 2, 3$	
si	$\sigma_i$	All	S	$\sigma_i$ normal stress global coord. system	$\sum_k (D_{ik} \epsilon_k)$
sxy	$\tau_{xy}$	All	S	$\tau_{xy}$ shear stress global coord. system	$\sum_k (D_{4k} \epsilon_k)$
si_t	$\sigma_{it}$	T	S	$\sigma_{it}$ time derivative of normal stress global coord. system	$\sum_k D_{ik} \dot{\epsilon}_{kt}$
sxy_t	$\tau_{xyt}$	T	S	$\tau_{xyt}$ time derivative of shear stress global coord. system	$\sum_k D_{4k} \dot{\epsilon}_{kt}$
sn	$\sigma_n$	All	S	$n$ th principal stress, $n = 1, 2, 3$	

TABLE 8-2: PLANE STRESS APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
sn <sub>i</sub>	$\sigma_{ni}$	All	S	<i>n</i> th principal stress component in the $x_i$ direction, $n = 1, 2, 3$	
tresca	$\sigma_{tresca}$	All	S	Tresca stress	$\max(\max( \sigma_1 - \sigma_2 ,  \sigma_2 - \sigma_3 ),  \sigma_1 - \sigma_3 )$
mises	$\sigma_{mises}$	All	S	von Mises stress	
Ta <sub>i</sub>	Ta <sub><i>i</i></sub>	All	B	Surface traction (force/area) in $x_i$ direction	$\begin{bmatrix} Ta_x \\ Ta_y \end{bmatrix} = \begin{bmatrix} \sigma_x & \tau_{xy} \\ \tau_{xy} & \sigma_y \end{bmatrix} \begin{bmatrix} n_x \\ n_y \end{bmatrix}$
F <sub>ig</sub>	F <sub><i>ig</i></sub>	All	S	Body load, in global $x_i$ direction	<p>If force/area</p> $\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \end{bmatrix}$ <p>If force/volume</p> $\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = th \begin{bmatrix} F_x \\ F_y \end{bmatrix}$
F <sub>ig</sub>	F <sub><i>ig</i></sub>	All	B	Edge load in global $x_i$ direction	<p>If force/length</p> $\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \end{bmatrix}$ <p>If force/area</p> $\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = th \begin{bmatrix} F_x \\ F_y \end{bmatrix}$
F <sub>ig</sub>	F <sub><i>ig</i></sub>	All	P	Point load in global $x_i$ direction	$\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \end{bmatrix}$

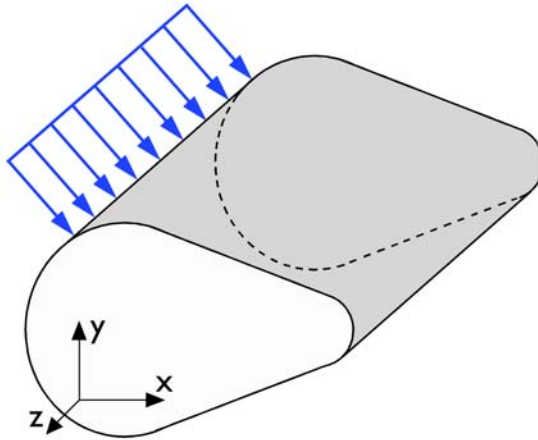
---

**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: `ps`), for example, `mises_ps`. (This does not apply to the dependent variables for the displacements.)

---

# The Plane Strain Application Mode

The Plane Strain application mode solves for the global displacements ( $u, v$ ) in the  $x$  and  $y$  directions. In a state of plane strain, the  $\epsilon_z$ ,  $\epsilon_{yz}$ , and  $\epsilon_{xz}$  components of the strain tensor are assumed to be zero.



## *Loads in a plane strain analysis.*

Loads in the  $x$  and  $y$  directions are allowed. The loads are assumed to be constant throughout the thickness of the material, but that thickness can vary in the  $x$  and  $y$  directions. The plane strain condition prevails in geometries that extend much farther in the  $z$  direction than in the  $x$  and  $y$  directions, or when the  $z$ -displacement is in some way restricted. The 2D geometry in a plane strain model represents a cross section that cuts a very long or infinite depth such that you can ignore any end effects. An example is a long tunnel along the  $z$ -axis where it is sufficient to study a unit-depth slice in the  $xy$ -plane. A plane strain model is sometimes also referred to as a *unit-depth model*.

## *Material Properties*

An additional material property for plane strain analysis is the geometry's thickness.

PARAMETER	VARIABLE	DESCRIPTION	MATERIAL MODEL
th	thickness	The geometry's thickness	All

## PDE Formulation

COMSOL Multiphysics formulates the application mode using the equilibrium equations described in general terms in the section “Implementation” on page 207.

### APPLICATION MODE PARAMETERS

The application mode parameters defining the loads, material, and constraints are explained in the section “Application Mode Descriptions” on page 210.

### Application Mode Variables

A large number of variables are available for use in expressions and for postprocessing. In addition to the variables listed below, almost all application mode parameters are available as variables. Some vary for different analyses as seen in the Analysis column.

ANALYSIS	ABBREVIATION
Static	S
Transient	T

Table 8-3 uses an index convention where a single index on  $u_i$  ( $u_i$ ) means that  $i$  runs over the global displacements ( $u, v$ ), whereas a single index on other names like  $s_i$  ( $\sigma_i$ ) means that  $i$  runs over the global spatial coordinates ( $x, y$ ). A single index on  $x_i$  means that  $i$  runs over the global spatial coordinates ( $x, y$ ).

TABLE 8-3: PLANE STRAIN APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
$u_i$	$u_i$	All	All	$x_i$ displacement	$u_i$
$u_{i\_t}$	$u_{it}$	T	All	$x_i$ velocity	$u_{it}$
disp	disp	All	All	Total displacement	$\sqrt{\sum_i (\text{real}(u_i))^2}$
$e_i$	$\varepsilon_i$	All	S	$\varepsilon_i$ normal strain global system	$\frac{\partial u_i}{\partial x_i}$
exy	$\varepsilon_{xy}$	All	S	$\varepsilon_{xy}$ shear strain global coord. system	$\frac{1}{2} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)$

TABLE 8-3: PLANE STRAIN APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
e <sub>i_t</sub>	$\epsilon_{it}$	T	S	$\epsilon_{it}$ velocity normal strain global system	$\frac{\partial u_{it}}{\partial x_i}$
exy <sub>t</sub>	$\epsilon_{xyt}$	T	S	$\epsilon_{xyt}$ velocity shear strain global coord. system	$\frac{1}{2} \left( \frac{\partial u_t}{\partial y} + \frac{\partial v_t}{\partial x} \right)$
en	$\epsilon_n$	All	S	$n$ th principal strain, $n = 1, 2, 3$	
eni	$\epsilon_{ni}$	All	S	$n$ th principal strain component in the $x_i$ direction, $n = 1, 2, 3$	
si	$\sigma_i$	All	S	$\sigma_i$ normal stress global coord. system	$\sum_{k=1,2,4} D_{ik}(\epsilon_k)$
sz	$\sigma_z$	All	S	$\sigma_z$ normal stress	$\sum_{k=1,2,4} D_{3k}(\epsilon_k)$
sxy	$\tau_{xy}$	All	S	$\tau_{xy}$ shear stress global coord. system	$\sum_{k=1,2,4} D_{4k}(\epsilon_k)$
si <sub>t</sub>	$\sigma_{it}$	T	S	$\sigma_{it}$ time derivative of normal stress global coord. system	$\sum_{k=1,2,4} D_{ik} \epsilon_{kt}$
sz <sub>t</sub>	$\sigma_z$	T	S	$\sigma_{zt}$ time derivative of normal stress	$\sum_{k=1,2,4} D_{3k} \epsilon_{kt}$
sxy <sub>t</sub>	$\tau_{xy}$	T	S	$\tau_{xy}$ shear stress global coord. system	$\sum_{k=1,2,4} D_{4k} \epsilon_{kt}$
sn	$\sigma_n$	All	S	$n$ th principal stress, $n = 1, 2, 3$	

TABLE 8-3: PLANE STRAIN APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
sn <sub>i</sub>	$\sigma_{ni}$	All	S	<i>n</i> th principal stress component in the $x_i$ direction, $n = 1, 2, 3$	
tresca	$\sigma_{tresca}$	All	S	Tresca stress	$\max(\max( \sigma_1 - \sigma_2 ,  \sigma_2 - \sigma_3 ,  \sigma_1 - \sigma_3 ))$
mises	$\sigma_{mises}$	All	S	von Mises stress	
Ta <sub>i</sub>	$Ta_i$	All	B	Surface traction (force/area) in $x_i$ direction	$\begin{bmatrix} Ta_x \\ Ta_y \end{bmatrix} = \begin{bmatrix} \sigma_x & \tau_{xy} \\ \tau_{xy} & \sigma_y \end{bmatrix} \begin{bmatrix} n_x \\ n_y \end{bmatrix}$
Fig	$F_{ig}$	All	S	Body load in global $x_i$ direction	<p>If force/area</p> $\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \end{bmatrix}$ <p>If force/volume</p> $\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = th \begin{bmatrix} F_x \\ F_y \end{bmatrix}$
Fig	$F_{ig}$	All	B	Edge load in global $x_i$ direction	<p>if force/length</p> $\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \end{bmatrix}$ <p>if force/area</p> $\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = th \begin{bmatrix} F_x \\ F_y \end{bmatrix}$
Fig	$F_{ig}$	All	P	Point load in global $x_i$ direction	$\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \end{bmatrix}$



---

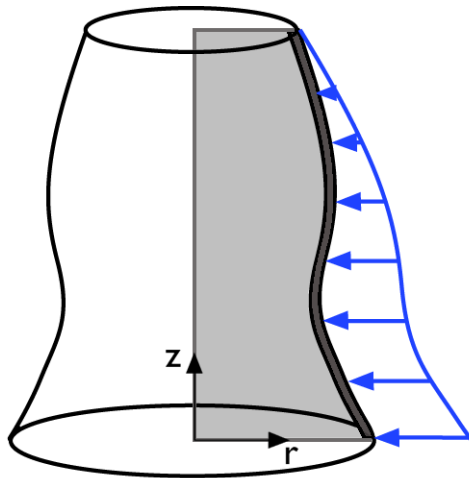
**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: pn), for example, `mises_pn`. (This does not apply to the dependent variables for the displacements.)

---

# The Axial Symmetry, Stress-Strain Application Mode

The Axial Symmetry, Stress-Strain application mode uses the cylindrical coordinates  $r$ ,  $\varphi$ , and  $z$ . It solves the equations for the global displacement ( $u$ ,  $w$ ) in the  $r$  and  $z$  directions. The displacement  $v$  in the  $\varphi$  direction together with the  $\tau_{r\varphi}$ ,  $\tau_{\varphi z}$ ,  $\gamma_{r\varphi}$ , and  $\gamma_{\varphi z}$  components of the stresses and strains are assumed to be zero. In this mode, loads are independent of  $\varphi$ , and it allows them only in the  $r$  and  $z$  directions.

You can view the domain where the equations are solved as the intersection between the original axisymmetric 3D solid and the half plane  $\varphi = 0$ ,  $r \geq 0$ . Therefore it is necessary to draw the geometry only in the half plane  $r \geq 0$ . Later on, recover the original 3D solid by rotating the 2D geometry about the  $z$ -axis (see the figure below).



*Loads in an axisymmetric stress-strain analysis. The modeling domain is the gray 2D section.*

The equilibrium equations in axial symmetry read

$$\begin{aligned}\frac{\partial \sigma_r}{\partial r} + \frac{\partial \tau_{rz}}{\partial z} + \frac{\sigma_r - \sigma_\theta}{r} + K_r &= 0 \\ \frac{\partial \tau_{rz}}{\partial r} + \frac{\partial \sigma_z}{\partial z} + \frac{\tau_{rz}}{r} + K_z &= 0\end{aligned}\tag{8-2}$$

The strain-displacement relationships for the axial symmetry case for small displacements are

$$\varepsilon_r = \frac{\partial u}{\partial r} \quad \varepsilon_\phi = \frac{u}{r} \quad \varepsilon_z = \frac{\partial w}{\partial z} \quad \gamma_{rz} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial r}$$

To avoid division by  $r$  in the equilibrium equations (which causes problems on the axis, where  $r = 0$ ), the COMSOL Multiphysics application mode transforms the equations. It multiplies the first equation by  $r^2$  and the second by  $r$ . This transformation is a natural choice because it appears in the principle of virtual work. Integrating over the volume, you must multiply the integrand by  $2\pi r$ . The application mode introduces and solves for a new dependent variable

$$\mathbf{uor} = \frac{u}{r}$$

which replaces the true radial displacement  $u$ .

---

**Note:**  $r = 0$  is the symmetry axis. In the Axial Symmetry, Stress-Strain application mode the  $r$ -axis is in the  $x$  direction (horizontal), and the  $z$ -axis is in the  $y$  direction (vertical).

---

### *PDE Formulation*

COMSOL Multiphysics formulates this application mode by using the equilibrium equation (see Equation 8-2) described in general terms in the section “Implementation” on page 207.

### **APPLICATION MODE PARAMETERS**

The application mode parameters defining the loads, material, and constraints are explained in the section “Application Mode Descriptions” on page 210.

## Application Mode Variables

A large number of variables are available for use in expressions and for postprocessing. In addition to the variables listed in Table 8-4, almost all application mode parameters are available as variables. Some vary for different analyses as you can see in the Analysis column.

ANALYSIS	ABBREVIATION
Static	S
Transient	T

TABLE 8-4: AXIAL SYMMETRY STRESS-STRAIN APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
uor	uor	All	All	$r$ displacement divided by $r$	uor
uaxi	uaxi	All	All	$r$ displacement	uor · $r$
w	w	All	All	$z$ displacement	w
uor_t	uor <sub>t</sub>	T	All	$r$ velocity divided by $r$	uor <sub>t</sub>
uaxi_t	uaxi <sub>t</sub>	T	All	$r$ velocity	uor <sub>t</sub> · $r$
w_t	w <sub>t</sub>	T	All	$z$ velocity	w <sub>t</sub>
disp	disp	All	All	Total displacement	$\sqrt{uaxi^2 + w^2}$
er	$\epsilon_r$	All	S	$\epsilon_r$ normal strain global system	uor + $\frac{\partial}{\partial r}(uor) \cdot r$
ez	$\epsilon_z$	All	S	$\epsilon_z$ normal strain global system	$\frac{\partial w}{\partial z}$
ephi	$\epsilon_\phi$	All	S	$\epsilon_\phi$ normal strain	uor
erz	$\epsilon_{rz}$	All	S	$\epsilon_{rz}$ shear strain global coord. system	$\frac{1}{2} \left( \frac{\partial}{\partial z}(uor) \cdot r + \frac{\partial w}{\partial r} \right)$
er_t	$\epsilon_{rt}$	T	S	$\epsilon_{rt}$ velocity normal strain global system	uor <sub>t</sub> + $\frac{\partial}{\partial r}(uor_t) \cdot r$
ez_t	$\epsilon_{zt}$	All	S	$\epsilon_{zt}$ velocity normal strain global system	$\frac{\partial w_t}{\partial z}$

TABLE 8-4: AXIAL SYMMETRY STRESS-STRAIN APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
ephi_t	$\varepsilon_{\phi t}$	All	S	$\varepsilon_{\phi t}$ velocity normal strain	$uor_t$
erz_t	$\varepsilon_{rzt}$	All	S	$\varepsilon_{rzt}$ shear strain global coord. system	$\frac{1}{2} \left( \frac{\partial}{\partial z} (uor_t) \cdot r + \frac{\partial w_t}{\partial r} \right)$
en	$\varepsilon_n$	All	S	$n$ th principal strain, $n = 1, 2, 3$	
eni	$\varepsilon_{ni}$	All	S	$n$ th principal strain component in the $x_i$ direction, $n = 1, 2, 3$	
sr	$\sigma_r$	All	S	$\sigma_r$ normal stress global coord. system	$\sum_k D_{1k}(\varepsilon_k)$
sphi	$\sigma_\phi$	All	S	$\sigma_\phi$ normal stress	$\sum_k D_{2k}(\varepsilon_k)$
sz	$\sigma_z$	All	S	$\sigma_z$ normal stress global coord. system	$\sum_k D_{3k}(\varepsilon_k)$
srz	$\tau_{rz}$	All	S	$\tau_{rz}$ shear stress global coord. system	$\sum_{k=1,2,4} D_{4k}(\varepsilon_k)$
sr_t	$\sigma_{rt}$	All	S	$\sigma_{rt}$ time derivative of normal stress global coord. system	$\sum_k D_{1k} \varepsilon_{kt}$
sphi_t	$\sigma_{\phi t}$	All	S	$\sigma_{\phi t}$ time derivative of normal stress	$\sum_k D_{2k} \varepsilon_{kt}$
sz_t	$\sigma_{zt}$	All	S	$\sigma_{zt}$ time derivative of normal stress global coord. system	$\sum_k D_{3k} \varepsilon_{kt}$

TABLE 8-4: AXIAL SYMMETRY STRESS-STRAIN APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
srz_t	$\tau_{rzt}$	All	S	$\tau_{rzt}$ time der. of shear stress global coord. system	$\sum_{k=1,2,4} D_{4k} \epsilon_{kt}$
sn	$\sigma_n$	All	S	$n$ th principal stress, $n = 1, 2, 3$	
sni	$\sigma_{ni}$	All	S	$n$ th principal stress component in the $x_i$ direction, $n = 1, 2, 3$	
tresca	$\sigma_{tresca}$	All	S	Tresca stress	$\max(\max( \sigma_1 - \sigma_2 ,  \sigma_2 - \sigma_3 ,  \sigma_1 - \sigma_3 ))$
mises	$\sigma_{mises}$	All	S	von Mises stress	
Tar, Taz	$Ta_r, Ta_z$	All	B	Surface traction (force/area) in $r$ and $z$ direction	$\begin{bmatrix} Ta_r \\ Ta_z \end{bmatrix} = \begin{bmatrix} \sigma_r & \tau_{rz} \\ \tau_{rz} & \sigma_z \end{bmatrix} \begin{bmatrix} n_r \\ n_z \end{bmatrix}$
Frg, Fzg	$F_{rg}, F_{zg}$	All	All	Body, edge, point load in global $r$ and $z$ directions	$\begin{bmatrix} F_{rg} \\ F_{zg} \end{bmatrix} = \begin{bmatrix} F_r \\ F_z \end{bmatrix}$

**Note:** To form the complete application mode variable names, add a suffix consisting of an underscore and the application mode name (default: axi), for example, `mises_axi`. (This does not apply to the dependent variables for the displacements.)

# Examples of Structural Mechanics Models

Two structural mechanics benchmark models show how to perform a linear static stress analysis using:

- A uniformly distributed horizontal load along an outer edge
- A gravity load

The examples are taken from a NAFEMS benchmark collection (Ref. 1).

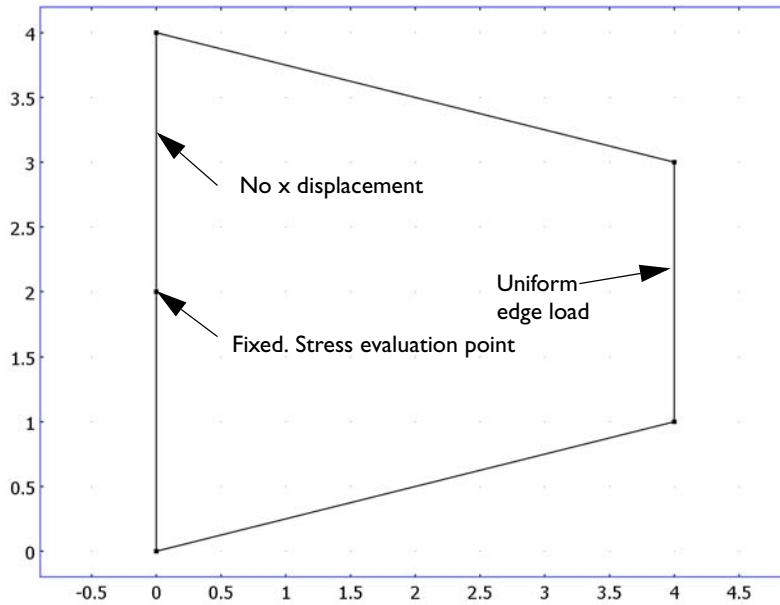
## *Tapered Membrane End Load*

---

The first example shows a 2D plane stress model of a membrane with a thickness of 0.1 m. The load is a uniformly distributed horizontal load of 10 MN/m (that is, a pressure of 100 MPa) along the right end. At the left end, there is no displacement in the  $x$  direction. Also, at a midpoint location, the left end is fixed in the  $y$  direction.

The model uses the following material properties:

- The material is isotropic.
- The Young's modulus (elasticity modulus) is  $210 \cdot 10^3$  MPa.
- The Poisson ratio is 0.3.



### *Modeling in COMSOL Multiphysics*

---

Using a Plane Stress application modes and a static analysis, it is straightforward to perform this stress analysis. The finite element model uses second-order triangular Lagrange elements. To show convergence toward the benchmark value, refine the mesh and recompute the solution twice.

### *Results*

---

The solution shows an  $x$ -direction stress at the point  $(0, 2)$  that is in good agreement with the benchmark target value of 61.3 MPa. Using the initial mesh, the COMSOL Multiphysics solution gives a value of 61.41 MPa. Two successive mesh refinements provide stress values of 61.36 MPa and 61.35 MPa.



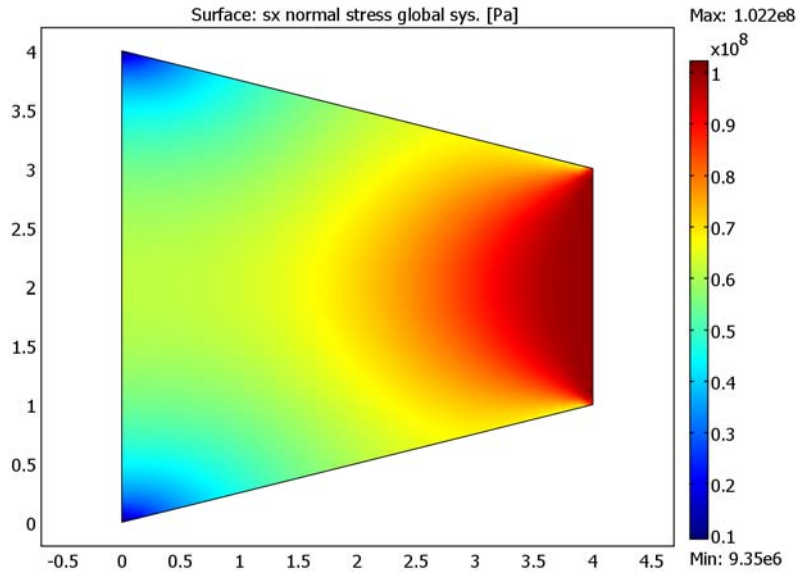


Figure 8-1: The  $x$ -direction stress distribution with a uniform edge load

---

**Model Library path:** COMSOL\_Multiphysics/Structural\_Mechanics/  
edge\_load\_2d

---

### *Modeling Using the Graphical User Interface*

---

#### **MODEL NAVIGATOR**

- 1 Select **2D** in the **Space dimension** list.
- 2 In the list of application modes, open the **COMSOL Multiphysics>Structural Mechanics** folder and then the **Plane Stress** node. Select **Static analysis**.
- 3 Click **OK**.

#### **GEOMETRY MODELING**

- 1 On the **Draw** menu, point to **Specify Objects** and then click **Line**.

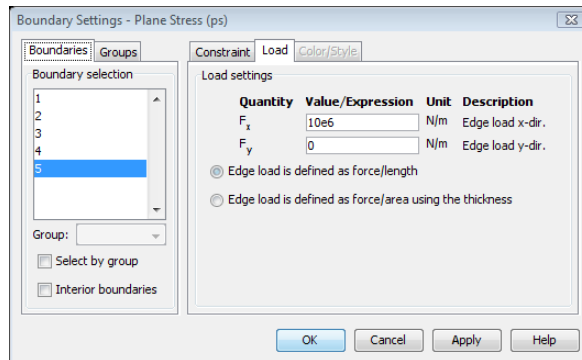
- 2 In the **Line** dialog box, type 0 4 4 0 0 in the **x** edit field and 0 1 3 4 0 in the **y** edit field.
- 3 Click **OK**.
- 4 Click the **Zoom Extents** button on the Main toolbar.
- 5 Click the **Coerce to Solid** button on the Draw toolbar.
- 6 On the **Draw** menu, point to **Specify Objects** and then click **Point**.
- 7 In the **Point** dialog box, type 0 in the **x** edit field and 2 in the **y** edit field.
- 8 Click **OK**.

The point is the location for the point constraint and also the benchmark value for the stress.

### PHYSICS SETTINGS

*Boundary and Point Conditions—Loads and Constraints*

- 1 From the **Physics** menu, choose **Boundary Settings**.
- 2 Select Boundaries 1 and 3 from the **Boundary selection** list.
- 3 Select the **R<sub>x</sub>** check box.
- 4 Click the **Load** tab.
- 5 Select Boundary 5 in the **Boundary selection** list.
- 6 Type 10e6 in the **F<sub>x</sub>** edit field.

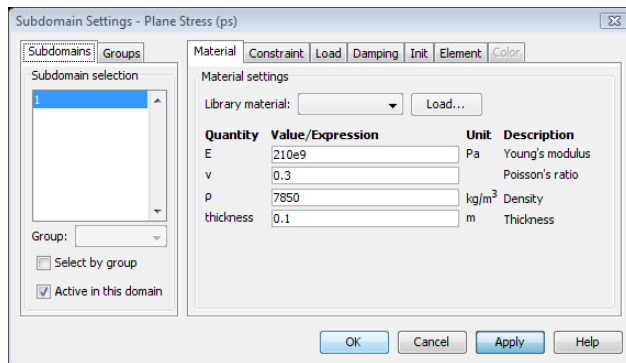


- 7 Click **OK**.
- 8 From the **Physics** menu, choose **Point Settings**.
- 9 Select Point 2 from the **Point selection** list.
- 10 Select the **R<sub>x</sub>** and **R<sub>y</sub>** check boxes.

11 Click **OK**.

### Subdomain Settings—Material Properties

- 1 From the **Physics** menu, choose **Subdomain Settings**.
- 2 Select Subdomain 1 from the **Subdomain selection** list.
- 3 Type 210e9 in the **E** edit field for the Young's modulus.
- 4 Type 0.3 in the **v** edit field for the Poisson's modulus.
- 5 Type 0.1 in the **thickness** edit field.



6 Click **OK**.

The other material properties are not used in this model.

## MESH GENERATION

Initialize an unstructured triangular mesh and refine it once:

- 1 Click the **Initialize Mesh** button on the Main toolbar.
- 2 Click the **Refine Mesh** button on the Main toolbar.

## COMPUTING THE SOLUTION

Click the **Solve** button on the Main toolbar.

## POSTPROCESSING AND VISUALIZATION

The default plot shows the von Mises stress. To plot the *x*-direction stress, use the following steps:

- 1 From the **Postprocessing** menu, select **Plot Parameters**.
- 2 In the **Plot Parameters** dialog box, click the **Surface** tab.

**3** On the **Surface Data** tab, select **sx normal stress global sys.** from the **Predefined quantities** list.

**4** Click **OK**.

To get a better look at the value of the  $x$ -direction stress at the point (0,2), use the **Cross-Section Plot Parameters** dialog box:

**1** From the **Postprocessing** menu, select **Cross-Section Plot Parameters**.

**2** Click the **Point** tab.

**3** Select **sx normal stress global sys.** from the **Predefined quantities** list.

**4** Type 0 in the **x** edit field and 2 in the **y** edit field.

**5** Click **OK**.

This plot shows a straight line representing the value of the  $x$ -direction stress at the point (0, 2). You can also click at that point in the results plot. That prints the numeric value in the message log.

### *Tapered Cantilever Gravity Load*

---

The second example shows another 2D plane stress model of a membrane with a thickness of 0.1 m. The load is a gravity load that acts in the negative  $y$  direction with an acceleration of  $9.81 \text{ m/s}^2$ . The left end boundary is fully fixed (no displacements).

The model uses the following material properties:

- The material is isotropic.
- The Young's modulus (elasticity modulus) is  $210 \cdot 10^3 \text{ MPa}$ .
- The Poisson ratio is 0.3.
- The density is  $7000 \text{ kg/m}^3$ .

### *Modeling in COMSOL Multiphysics*

---

Using a Plane Stress application modes and a static analysis, it is straightforward to perform this stress analysis. You enter the gravity load as force/volume. COMSOL Multiphysics then computes the load using the thickness of the material. The finite element model uses second-order triangular Lagrange elements. For postprocessing, use a cross-section plot to show the value of the shear stress at the location (0, 2).

## Results

---

The solution shows a shear stress ( $s_{xy}$ ) at the point (0, 2) that is in good agreement with the benchmark target value of  $-0.200$  MPa. Using the initial mesh, the COMSOL Multiphysics solution gives a value of  $-0.199$  MPa.

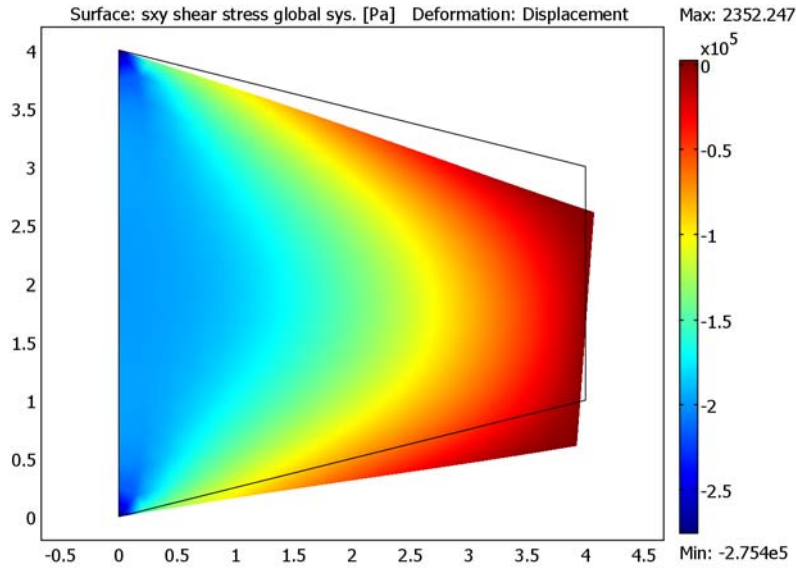


Figure 8-2: The shear stress and the displacement (exaggerated) from a gravity load.

---

**Model Library path:** COMSOL\_Multiphysics/Structural\_Mechanics/  
gravity\_load\_2d

---

## Modeling Using the Graphical User Interface

---

### MODEL NAVIGATOR

- 1 Select **2D** from the **Space dimension** list.
- 2 In the list of application modes, open the **COMSOL Multiphysics>Structural Mechanics** folder and then the **Plane Stress** node. Select **Static analysis**.
- 3 Click **OK**.

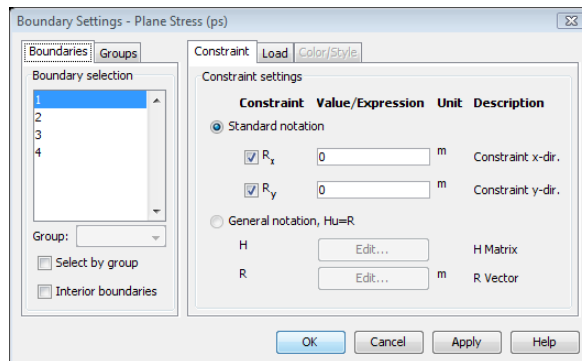
## GEOMETRY MODELING

- 1 On the **Draw** menu, point to **Specify Objects** and then click **Line**.
- 2 In the **Line** dialog box, type 0 4 4 0 0 in the **x** edit field and 0 1 3 4 0 in the **y** edit field.
- 3 Click **OK**.
- 4 Click the **Zoom Extents** button on the Main toolbar.
- 5 Click the **Coerce to Solid** button on the Draw toolbar.

## PHYSICS SETTINGS

### *Boundary Conditions—Constraints*

- 1 From the **Physics** menu, choose **Boundary Settings**.
- 2 Select Boundary 1 in the **Boundary selection** list.
- 3 Select the **R<sub>x</sub>** and **R<sub>y</sub>** check boxes.



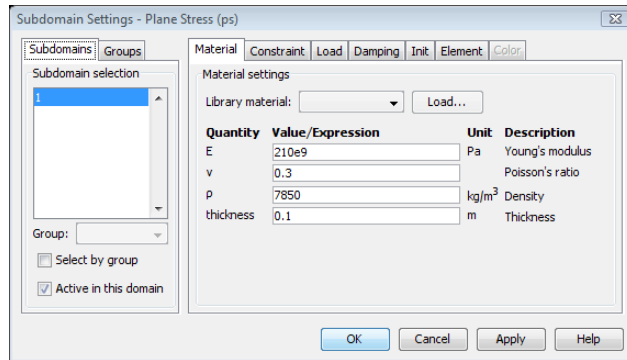
- 4 Click **OK**.

### *Subdomain Settings—Material Properties and Loads*

- 1 From the **Physics** menu, choose **Subdomain Settings**.
- 2 Select Subdomain 1 in the **Subdomain selection** list.
- 3 Type 210e9 in the **E** edit field for the Young's modulus.
- 4 Type 0.3 in the **v** edit field for the Poisson's ratio.

5 Type 0.1 in the **thickness** edit field.

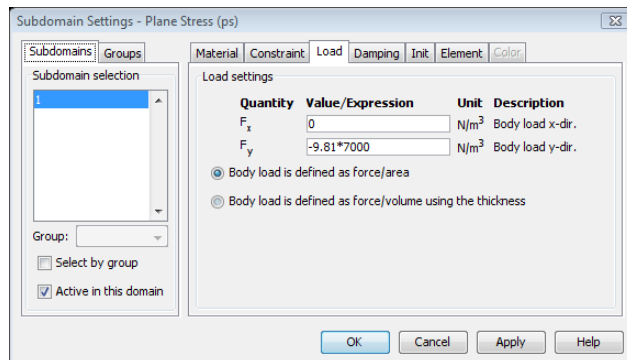
You can also type 7000 in the  $\rho$  edit field for the density, but the density value from this edit field is not used in the static analysis. The gravity appears in the specification of the gravity load below.



6 Click the **Load** tab.

7 Type  $-9.81 \cdot 7000$  in the  $F_y$  edit field for the  $y$ -direction body load.

8 Click the **Body load is defined as force/volume using the thickness** button.



9 Click **OK**.

## MESH GENERATION

Click the **Initialize Mesh** button on the Main toolbar.

## COMPUTING THE SOLUTION

Click the **Solve** button on the Main toolbar.

## POSTPROCESSING AND VISUALIZATION

The default plot shows the von Mises stress. To plot the shear stress and also the deformed shape (displacements), use the following steps:

- 1 From the **Postprocessing** menu, select **Plot Parameters**.
- 2 In the **Plot Parameters** dialog box, click the **Surface** tab.
- 3 On the **Surface Data** tab, select **sxy shear stress global sys.** from the **Predefined quantities** list.
- 4 Click the **Deform** tab.
- 5 Select the **Deformed shape plot** check box. The deformation data is the  $x$  and  $y$  displacements by default.
- 6 Click **OK**.

The deformed shape plot uses a scaling factor to clearly show the deformation. The actual displacements are small compared to the model geometry. Click the **Deform** tab too see the scale factor or to enter a different scale factor.

To get a better look at the value of the shear stress at the point (0, 2), use the **Cross-Section Plot Parameters** dialog box:

- 1 From the **Postprocessing** menu, select **Cross-Section Plot Parameters**.
- 2 Click the **Point** tab.
- 3 Select **sxy shear stress global sys.** from the **Predefined quantities** list.
- 4 Type 0 in the  $x$  edit field and 2 in the  $y$  edit field.
- 5 Click **OK**.

The plot shows a straight line at the value of the shear stress at (0, 2). You can also click at that point in the results plot. That prints the numeric value in the message log.

### *Reference*

---

1. D. Hitchings, A. Kamoulakos, G. A. O. Davies: *Linear Statics Benchmarks Vol. 1*, NAFEMS Ltd., Glasgow, 1987.



## PDE Modes for Equation-Based Modeling

This chapter describes the use of the PDE modes for equation-based modeling. A step-by-step example illustrates the use of the PDE modes for solving Poisson's equation.

# The PDE Modes

The PDE modes are application modes for equation-based modeling. They support three types of PDE formulation:

- *Coefficient form* for linear or almost linear PDEs, explained in detail in the section “Using the Coefficient Form PDEs” on page 247.
- *General form* for nonlinear PDEs. You find a detailed discussion about the general form in the section “Using the General Form PDE” on page 257.
- *Weak form* using the weak formulation of the PDE for maximum flexibility. For more information about the weak form, see “The Weak Form” on page 349.

In addition, a number of specialized instances of these PDE formulations provide interfaces for a number of classical PDEs. See “Classical PDEs” on page 268 for more information.

The following table lists the available PDE modes with their default application mode names:

TABLE 9-1: DEFAULT APPLICATION MODE NAMES FOR THE PDE MODES

PDE MODE	APPLICATION MODE NAME
PDE, Coefficient Form	c
PDE, General Form	g
Weak Form, Subdomain	w
Weak Form, Boundary	wb
Weak Form, Edge	we
Weak Form, Point	wp
Convection-Diffusion Equation	cdeq
Heat Equation	hteq
Helmholtz Equation	hzeq
Laplace’s Equation	lpeq
Poisson’s Equation	poeq
Schrödinger Equation	scheq
Wave Equation	waeq

# Using the PDE Modes

## *Starting a Model Using a PDE Mode*

---

Do as follows to create a new model using one of the PDE modes:

- 1 Start COMSOL Multiphysics or click **New** in the COMSOL Multiphysics window.
- 2 In the **Model Navigator** make a selection from the **Space dimension** list.
- 3 Go to the folder **Application Modes>COMSOL Multiphysics>PDE Modes**.
- 4 Open either the **PDE, Coefficient Form** folder, the **PDE, General Form** folder, or one of the weak form folders, and make an optional further selection between the available analysis types.

Alternatively, open the **Classical PDEs** folder to select from a list of classical PDEs, which are all special cases of the coefficient form formulation.

- 5 Click **OK**.

### **SPECIFYING A SYSTEM OF PDES**

COMSOL Multiphysics allows the creation of PDEs with more than one dependent variable. To add variables, follow Steps 1 through 4 above and then enter all the variable names as space-separated entries in the **Dependent variables** edit field before clicking **OK**. (For the classical PDEs, you can change the name of the dependent variable but not add new ones.) The number of equations in the model equals the number of dependent variables. For example, to start a model with three dependent variables  $u_1$ ,  $u_2$ , and  $u_3$ , enter:

u1 u2 u3

You can also couple several scalar PDEs using a multiphysics approach.

## *Using the Coefficient Form PDEs*

---

The coefficient form PDE mode covers many well-known PDEs. This section covers the formulation and settings pertaining to the coefficient form, as well as the general PDE terminology used in COMSOL Multiphysics.

## THE SCALAR COEFFICIENT FORM EQUATION

A single dependent variable  $u$  is an unknown function on the computational domain. COMSOL Multiphysics determines it by solving the PDE problem that you specify. In coefficient form, the PDE problem reads

$$\begin{cases} e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} + \nabla \cdot (-c \nabla u - \alpha u + \gamma) + \beta \cdot \nabla u + a u = f & \text{in } \Omega \\ \mathbf{n} \cdot (c \nabla u + \alpha u - \gamma) + q u = g - h^T \boldsymbol{\mu} & \text{on } \partial\Omega \\ h u = r & \text{on } \partial\Omega \end{cases} \quad (9-1)$$

where

- $\Omega$  is the computational domain—the union of all subdomains
- $\partial\Omega$  is the domain boundary
- $\mathbf{n}$  is the outward unit normal vector on  $\partial\Omega$

The first equation in the list above is the PDE, which must be satisfied in  $\Omega$ . The second and third equations are the boundary conditions, which must hold on  $\partial\Omega$ . The second equation is a *generalized Neumann* boundary condition, whereas the third equation is a *Dirichlet* boundary condition. This nomenclature and the second equation above deviate slightly from traditional usage in potential theory where a Neumann condition usually refers to the case  $q = 0$ . The generalized Neumann condition is also called a *mixed boundary condition* or a *Robin boundary condition*. In finite element terminology, Neumann boundary conditions are called *natural boundary conditions* because they do not occur explicitly in the weak form of the PDE problem. Dirichlet conditions are called *essential boundary conditions* because they restrict the trial space. Dirichlet boundary conditions often represent *constraints*.

This manual uses the following conventions:

- The symbol  $\nabla$  is the vector differential operator (gradient), defined as

$$\nabla = \left( \frac{\partial}{\partial x_1}, \dots, \frac{\partial}{\partial x_n} \right)$$

The space coordinates are denoted  $x_1, \dots, x_n$ , where  $n$  represents the number of space dimensions.

- The symbol  $\Delta$  stands for the Laplace operator

$$\frac{\partial^2}{\partial x_1^2} + \dots + \frac{\partial^2}{\partial x_n^2}$$

- $\nabla \cdot (c \nabla u)$  means

$$\frac{\partial}{\partial x_1} \left( c \frac{\partial u}{\partial x_1} \right) + \dots + \frac{\partial}{\partial x_n} \left( c \frac{\partial u}{\partial x_n} \right)$$

- $\beta \cdot \nabla u$  means

$$\beta_1 \frac{\partial u}{\partial x_1} + \dots + \beta_n \frac{\partial u}{\partial x_n}$$

where  $\beta_1, \dots, \beta_n$  are the components of the vector  $\beta$ .

Within COMSOL Multiphysics, you specify the coefficients  $c$ ,  $\alpha$ ,  $\gamma$ ,  $\beta$ ,  $\alpha$ ,  $q$ , and  $h$ , and the terms  $f$ ,  $g$ , and  $r$ . They can all be functions of the spatial coordinates.

- A PDE is *linear* when the coefficients depend only on the spatial coordinates (or are constants).
- A PDE is *nonlinear* if the coefficients depend on  $u$  or its derivatives (the components of  $\nabla u$ ).
- All the coefficients in the above equation are scalars except  $\alpha$ ,  $\beta$ , and  $\gamma$ , which are vectors with  $n$  components. The coefficient  $c$  can alternatively be an  $n$ -by- $n$  matrix to model anisotropic materials. For more information see “Modeling Anisotropic Materials” on page 221 in the *COMSOL Multiphysics User’s Guide*.

The  $e_a$  coefficient in Equation 9-1 is a scalar or a matrix for time-dependent systems called the *mass matrix* (or mass coefficient). The  $d_a$  coefficient represents a damping term (however, if  $e_a = 0$ , then  $d_a$  is often called the mass coefficient). See “Solving Time-Dependent Problems” on page 264 for more information on time-dependent problems.

#### *Interpreting Boundary Conditions*

The formulation of the boundary conditions imposes both Dirichlet and Neumann conditions. This combination is possible because of a new dependent variable  $\mu$ , which is defined only on the boundary. The unknown variable  $\mu$  is called a *Lagrange multiplier*. Often you can reformulate boundary conditions without Lagrange multipliers. In structural mechanics problems the Lagrange multiplier equals the reaction forces on the boundary. The factor  $h^T$  in the Neumann boundary condition is the transpose of  $h$ . If  $h$  is a scalar, then  $h^T = h$ .

How do the Lagrange multipliers relate to the conventional formulation with either a Dirichlet or a Neumann boundary condition?

- First, assume that  $h = 1$ . Then the Dirichlet condition is  $u = r$ . The Neumann condition becomes:

$$\mathbf{n} \cdot (c\nabla u + \alpha u - \gamma) + qu = g - \mu$$

The Lagrange multiplier,  $\mu$ , adjusts so as to satisfy the requested Dirichlet condition. Specifying a nonzero  $g$  or  $q$  changes the value of the Lagrange multiplier but does not affect the actual solution  $u$ . Therefore, this equation can usually be ignored, leaving a pure Dirichlet condition.

- Second, assume that  $h = 0$  and  $r = 0$ . Then the Dirichlet condition reads  $0 = 0$ , and the Neumann condition is

$$\mathbf{n} \cdot (c\nabla u + \alpha u - \gamma) + qu = g$$

This is the generalized Neumann condition without a Lagrange multiplier. For more on the Lagrange multiplier formulation, see “System for Two Variables in the General Form” on page 259.

The vector  $\Gamma = -c\nabla u - \alpha u + \gamma$  is the *flux vector*. In transport equations its first term describes diffusion, the second term describes convection with a velocity  $-\alpha$ , and the third term  $\gamma$  is a source term.

In certain applications  $\Gamma$  can be discontinuous across interior boundaries. There can be a jump in the normal component of  $\Gamma$  across such a boundary. For instance, denote the two adjacent subdomains as 1 and 2, and let  $\Gamma_i$  and  $\mathbf{n}_i$  be the values of  $\Gamma$  and  $\mathbf{n}$  from the two subdomains. Then you can state the jump condition as the Neumann condition

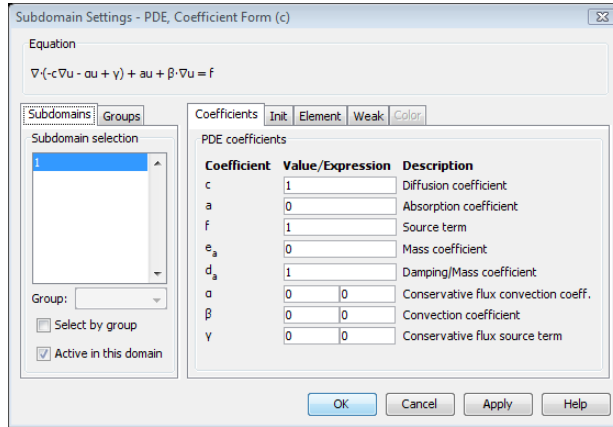
$$-\mathbf{n}_1 \cdot \Gamma_1 - \mathbf{n}_2 \cdot \Gamma_2 + qu = g - h^T \mu$$

where  $\mathbf{n}_1$  is the outward normal from Subdomain 1, and  $\mathbf{n}_2$  is the outward normal from Subdomain 2 so that  $\mathbf{n}_1 = -\mathbf{n}_2$ . At the same time there is a Dirichlet condition  $hu = r$ . (Let  $h = r = 0$  to discard the Dirichlet condition and the Lagrange multiplier  $\mu$ .) To specify such conditions on interior boundaries, select the **Enable interior boundaries** check box in the **Boundary Settings** dialog box.

The rest of this section handles stationary problems where  $\partial u / \partial t = 0$  and transient effects have vanished.

### Specifying PDE Coefficients on Subdomains

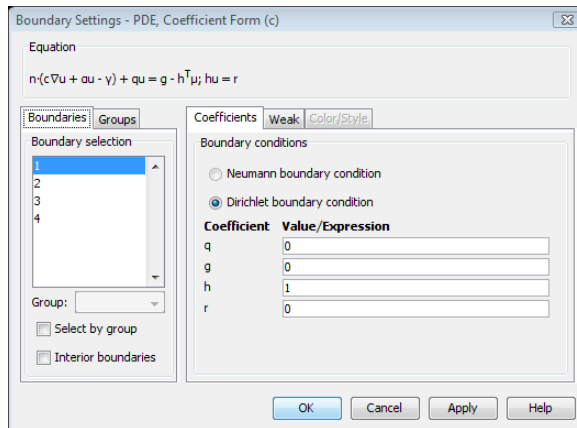
To specify PDE coefficients, open the **Subdomain Settings** dialog box for a stationary, scalar 2D PDE problem in coefficient form. From the **Physics** menu, choose **Subdomain Settings**.



The default values correspond to Poisson's equation with a source value of 1.

### Specifying Boundary Conditions

To specify boundary conditions, open the **Boundary Settings** dialog box. From the **Physics** menu, choose **Boundary Settings**.



The default values correspond to homogeneous Dirichlet boundary conditions.

To specify boundary conditions:

- 1 Make a selection from the **Boundary selection** list.
- 2 Select a Neumann or Dirichlet boundary condition type.
- 3 Type the values for the coefficients that are active for the boundary condition type.

**Note:** The  $c$ ,  $\alpha$ , and  $\gamma$  coefficients in the equations for the boundary conditions come from the PDE specification in the **Subdomain Settings** dialog box.

### THE COEFFICIENT FORM EQUATION SYSTEM

With two independent variables  $u_1$  and  $u_2$ , the stationary PDE problem in coefficient form results in the following equation system:

$$e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} - \nabla \cdot (c \nabla u + \alpha u - \gamma) + \beta \cdot \nabla u + \alpha u = f$$

where  $u = (u_1, u_2)$ . The mass term is defined as

$$e_a \frac{\partial^2 u}{\partial t^2} = \begin{bmatrix} e_{a11} & e_{a12} \\ e_{a21} & e_{a22} \end{bmatrix} \begin{bmatrix} \frac{\partial^2 u_1}{\partial t^2} \\ \frac{\partial^2 u_2}{\partial t^2} \end{bmatrix} = \begin{bmatrix} e_{a11} \frac{\partial^2 u_1}{\partial t^2} + e_{a12} \frac{\partial^2 u_2}{\partial t^2} \\ e_{a21} \frac{\partial^2 u_1}{\partial t^2} + e_{a22} \frac{\partial^2 u_2}{\partial t^2} \end{bmatrix}$$

Similarly, the damping term is

$$d_a \frac{\partial u}{\partial t} = \begin{bmatrix} d_{a11} & d_{a12} \\ d_{a21} & d_{a22} \end{bmatrix} \begin{bmatrix} \frac{\partial u_1}{\partial t} \\ \frac{\partial u_2}{\partial t} \end{bmatrix} = \begin{bmatrix} d_{a11} \frac{\partial u_1}{\partial t} + d_{a12} \frac{\partial u_2}{\partial t} \\ d_{a21} \frac{\partial u_1}{\partial t} + d_{a22} \frac{\partial u_2}{\partial t} \end{bmatrix}$$

However, if  $e_a = 0$ , then  $d_a$  is often called the mass coefficient.

The diffusive flux is defined as



$$\begin{aligned}
c \nabla u &= \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \nabla \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} \nabla u_1 \\ \nabla u_2 \end{bmatrix} = \begin{bmatrix} c_{11} \nabla u_1 + c_{12} \nabla u_2 \\ c_{21} \nabla u_1 + c_{22} \nabla u_2 \end{bmatrix} \\
&= \begin{bmatrix} c_{11} \begin{bmatrix} \frac{\partial u_1}{\partial x} \\ \frac{\partial u_1}{\partial y} \end{bmatrix} + c_{12} \begin{bmatrix} \frac{\partial u_2}{\partial x} \\ \frac{\partial u_2}{\partial y} \end{bmatrix} \\ c_{21} \begin{bmatrix} \frac{\partial u_1}{\partial x} \\ \frac{\partial u_1}{\partial y} \end{bmatrix} + c_{22} \begin{bmatrix} \frac{\partial u_2}{\partial x} \\ \frac{\partial u_2}{\partial y} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} c_{11} \frac{\partial u_1}{\partial x} + c_{12} \frac{\partial u_2}{\partial x} \\ c_{11} \frac{\partial u_1}{\partial y} + c_{12} \frac{\partial u_2}{\partial y} \\ c_{21} \frac{\partial u_1}{\partial x} + c_{22} \frac{\partial u_2}{\partial x} \\ c_{21} \frac{\partial u_1}{\partial y} + c_{22} \frac{\partial u_2}{\partial y} \end{bmatrix} = \begin{bmatrix} [cu1x] \\ [cu1y] \\ [cu2x] \\ [cu2y] \end{bmatrix}
\end{aligned}$$

where  $\nabla u_1$  and  $\nabla u_2$  are column vectors. The flux matrix or flux tensor is a column vector in this presentation. For anisotropic materials, the components  $c_{11}$ ,  $c_{12}$ ,  $c_{21}$ , and  $c_{22}$  can be matrices as described above for the one-variable coefficient form PDE. In this case, the diffusive flux reads

$$\begin{aligned}
c \nabla u &= \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} \nabla u_1 \\ \nabla u_2 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} c_{1111} & c_{1112} \\ c_{1121} & c_{1122} \end{bmatrix} \begin{bmatrix} \nabla u_1 \\ \nabla u_2 \end{bmatrix} \\ \begin{bmatrix} c_{2111} & c_{2112} \\ c_{2121} & c_{2122} \end{bmatrix} \begin{bmatrix} \nabla u_1 \\ \nabla u_2 \end{bmatrix} \end{bmatrix} \\
&= \begin{bmatrix} \begin{bmatrix} c_{1111} & c_{1112} \\ c_{1121} & c_{1122} \end{bmatrix} \nabla u_1 + \begin{bmatrix} c_{1211} & c_{1212} \\ c_{1221} & c_{1222} \end{bmatrix} \nabla u_2 \\ \begin{bmatrix} c_{2111} & c_{2112} \\ c_{2121} & c_{2122} \end{bmatrix} \nabla u_1 + \begin{bmatrix} c_{2211} & c_{2212} \\ c_{2221} & c_{2222} \end{bmatrix} \nabla u_2 \end{bmatrix}
\end{aligned}$$

The conservative convective flux is defined as

$$\alpha u = \alpha \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \alpha_{12} \\ \alpha_{21} & \alpha_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \alpha_{111} & \alpha_{121} \\ \alpha_{112} & \alpha_{122} \end{bmatrix} \\ \begin{bmatrix} \alpha_{211} & \alpha_{221} \\ \alpha_{212} & \alpha_{222} \end{bmatrix} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \alpha_{111} \\ \alpha_{112} \end{bmatrix} u_1 + \begin{bmatrix} \alpha_{121} \\ \alpha_{122} \end{bmatrix} u_2 \\ \begin{bmatrix} \alpha_{211} \\ \alpha_{212} \end{bmatrix} u_1 + \begin{bmatrix} \alpha_{221} \\ \alpha_{222} \end{bmatrix} u_2 \end{bmatrix}$$

Here the third index,  $k$ , of  $\alpha_{ijk}$  corresponds to the space coordinate suffixes  $x$  and  $y$ .

The conservative flux source is defined as

$$\gamma = \begin{bmatrix} \gamma_{11} \\ \gamma_{12} \\ \gamma_{21} \\ \gamma_{22} \end{bmatrix}$$

Here the second index,  $j$ , of  $\gamma_{ij}$  denotes the space coordinate suffixes for  $x$  and  $y$ .

For the flux terms the divergence operator works on each row separately. To illustrate this, consider the divergence of the conservative flux source

$$\nabla \cdot \gamma = \nabla \cdot \begin{bmatrix} \gamma_{11} \\ \gamma_{12} \\ \gamma_{21} \\ \gamma_{22} \end{bmatrix} = \begin{bmatrix} \nabla \cdot \begin{bmatrix} \gamma_{11} \\ \gamma_{12} \end{bmatrix} \\ \nabla \cdot \begin{bmatrix} \gamma_{21} \\ \gamma_{22} \end{bmatrix} \end{bmatrix}$$

The convection term is defined as

$$\beta \cdot \nabla u = \begin{bmatrix} \beta_{11} & \beta_{12} \\ \beta_{21} & \beta_{22} \end{bmatrix} \cdot \begin{bmatrix} \nabla u_1 \\ \nabla u_2 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \beta_{111} & \beta_{121} \\ \beta_{112} & \beta_{122} \end{bmatrix} \\ \begin{bmatrix} \beta_{211} & \beta_{221} \\ \beta_{212} & \beta_{222} \end{bmatrix} \end{bmatrix} \cdot \begin{bmatrix} \nabla u_1 \\ \nabla u_2 \end{bmatrix} = \begin{bmatrix} \begin{bmatrix} \beta_{111} \\ \beta_{112} \end{bmatrix} \cdot \nabla u_1 + \begin{bmatrix} \beta_{121} \\ \beta_{122} \end{bmatrix} \cdot \nabla u_2 \\ \begin{bmatrix} \beta_{211} \\ \beta_{212} \end{bmatrix} \cdot \nabla u_1 + \begin{bmatrix} \beta_{221} \\ \beta_{222} \end{bmatrix} \cdot \nabla u_2 \end{bmatrix}$$

The variable names for these components are `beu1` and `beu2`.

The absorption term is defined as

$$\alpha u = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} a_{11}u_1 + a_{12}u_2 \\ a_{21}u_1 + a_{22}u_2 \end{bmatrix}$$

The variable names for these components are `au1` and `au2`.

The source term is defined as

$$f = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$$

The variable names for these components are `f1` and `f2`.

*The Boundary Condition Terms*

The Dirichlet boundary condition, in expanded form, reads

$$\begin{bmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \end{bmatrix}$$

If you choose the Dirichlet condition, you also get the generalized Neumann boundary condition, which reads

$$\mathbf{n} \cdot (c \nabla u + \alpha u - \gamma) + qu = g - \mathbf{h}^T \boldsymbol{\mu}$$

The normal vector  $\mathbf{n} = (n_x, n_y)$  operates on the flux vector in the same way as the divergence operator as explained earlier. If  $\mathbf{h}$  has full rank (as in the default identity matrix, for example) only the constraints from the Dirichlet condition are active.

If you choose the Neumann condition, you get only the boundary condition

$$\mathbf{n} \cdot (c \nabla u + \alpha u - \gamma) + qu = g$$

The normal component of the diffusive flux is defined as

$$\mathbf{n} \cdot c \nabla u = \mathbf{n} \cdot \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \begin{bmatrix} \nabla u_1 \\ \nabla u_2 \end{bmatrix} = \begin{bmatrix} \mathbf{n} \cdot (c_{11} \nabla u_1 + c_{12} \nabla u_2) \\ \mathbf{n} \cdot (c_{21} \nabla u_1 + c_{22} \nabla u_2) \end{bmatrix}$$

The normal component of the conservative convective flux is defined as

$$\mathbf{n} \cdot \alpha u = \mathbf{n} \cdot \begin{bmatrix} \begin{bmatrix} \alpha_{111} \\ \alpha_{112} \end{bmatrix} u_1 + \begin{bmatrix} \alpha_{121} \\ \alpha_{122} \end{bmatrix} u_2 \\ \begin{bmatrix} \alpha_{211} \\ \alpha_{212} \end{bmatrix} u_1 + \begin{bmatrix} \alpha_{221} \\ \alpha_{222} \end{bmatrix} u_2 \end{bmatrix} = \begin{bmatrix} (n_x, n_y) \cdot \begin{bmatrix} \alpha_{111} \\ \alpha_{112} \end{bmatrix} u_1 + \begin{bmatrix} \alpha_{121} \\ \alpha_{122} \end{bmatrix} u_2 \\ (n_x, n_y) \cdot \begin{bmatrix} \alpha_{211} \\ \alpha_{212} \end{bmatrix} u_1 + \begin{bmatrix} \alpha_{221} \\ \alpha_{222} \end{bmatrix} u_2 \end{bmatrix}$$

The normal component of the conservative flux source is defined as

$$\mathbf{n} \cdot \gamma = (n_x, n_y) \cdot \begin{bmatrix} \gamma_{11} \\ \gamma_{12} \\ \gamma_{21} \\ \gamma_{22} \end{bmatrix} = \begin{bmatrix} (n_x, n_y) \cdot \begin{bmatrix} \gamma_{11} \\ \gamma_{12} \end{bmatrix} \\ (n_x, n_y) \cdot \begin{bmatrix} \gamma_{21} \\ \gamma_{22} \end{bmatrix} \end{bmatrix}$$

The boundary absorption term is defined as

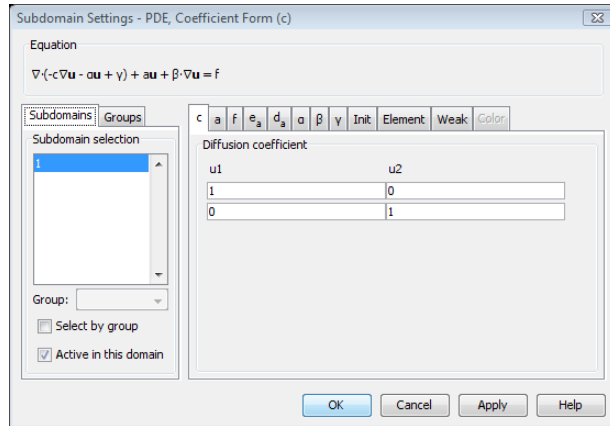
$$qu = \begin{bmatrix} q_{11} & q_{12} \\ q_{21} & q_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} q_{11}u_1 + q_{12}u_2 \\ q_{21}u_1 + q_{22}u_2 \end{bmatrix}$$

The boundary source term is defined as

$$g = \begin{bmatrix} g_1 \\ g_2 \end{bmatrix}$$

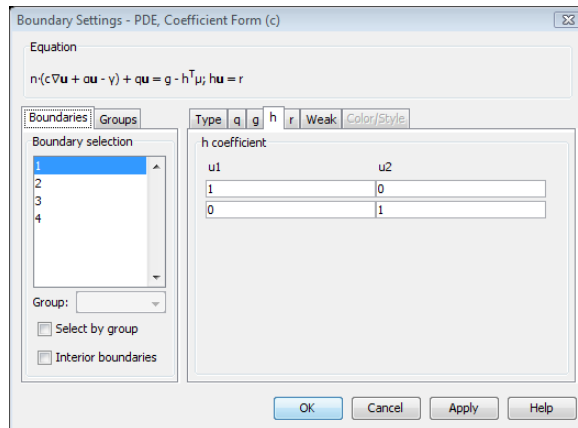
### Specifying PDE Coefficients on Subdomains

The following figure shows the **Subdomain Settings** dialog box for a coefficient form, two-variable stationary 2D problem.



### Specifying Boundary Conditions

The **Boundary Settings** dialog box looks like this:



### Using the General Form PDE

The general form PDE provides a framework for specification of PDEs that can be nonlinear.

## THE SCALAR GENERAL FORM EQUATION

Use the *general form* for nonlinear PDEs. Assuming that you are working with a single dependent variable  $u$ , then a stationary problem in general form reads

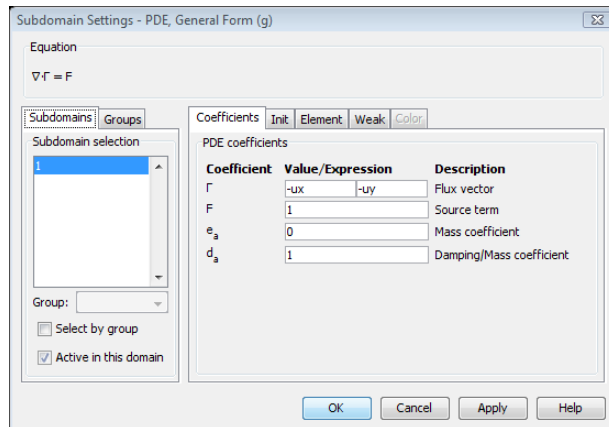
$$\begin{cases} \nabla \cdot \Gamma = F & \text{in } \Omega \\ -\mathbf{n} \cdot \Gamma = G + \left(\frac{\partial R}{\partial u}\right)^T \mu & \text{on } \partial\Omega \\ 0 = R & \text{on } \partial\Omega \end{cases}$$

The first equation is the PDE. The second and third equations are the Neumann and Dirichlet boundary conditions, respectively. For information on the time-dependent general form equation, see “Solving Time-Dependent Problems” on page 264.

The terms  $\Gamma$ ,  $F$ ,  $G$ , and  $R$  are coefficients. They can be functions of the spatial coordinates, the solution  $u$ , and the space derivatives of  $u$ . The coefficients  $F$ ,  $G$ , and  $R$  are scalar, whereas  $\Gamma$  is the *flux vector*. The superscript “ $T$ ” in the Neumann boundary condition denotes the transpose. The variable  $\mu$  is the Lagrange multiplier.

### Specifying PDE Coefficients on Subdomains

The following image shows the **Subdomain Settings** dialog box for a general form, scalar, stationary 2D problem.



### Specifying Boundary Conditions

The **Boundary Settings** dialog box entries are similar to the one for the coefficient form PDEs, but in general form you specify only the  $g$  and  $r$  coefficients.

### THE GENERAL FORM EQUATION SYSTEM

In the case of several dependent variables  $u_1, u_2, \dots, u_N$ , the following system of equations represents a stationary problem in the general form:

$$\left\{ \begin{array}{ll} \nabla \cdot \Gamma_l = F_l & \text{in } \Omega \\ -\mathbf{n} \cdot \Gamma_l = G_l + \frac{\partial R_m}{\partial u_l} \mu_m & \text{on } \partial\Omega \\ 0 = R_m & \text{on } \partial\Omega \end{array} \right.$$

The equation index  $l$  ranges from 1 to  $N$ , while the constraint index  $m$  ranges from 1 to  $M$ . This discussion uses the summation convention.  $F_l, G_l$ , and  $R_m$  are scalars, whereas  $\Gamma_l$  is a vector. In this case there are several Lagrange multipliers:  $\mu_1, \mu_2, \dots, \mu_M$ .

For a more compact form, let  $u$  be a vector with components  $u_k$ , let  $\Gamma$  be a vector with components  $\Gamma_l$ , and so on. Then the system of equations takes on the same form as given above for a single dependent variable.

It is possible to rewrite the system to introduce the components  $\Gamma_{lj}$  of the vector  $\Gamma_l$  and the components  $n_j$  of the normal vector  $\mathbf{n}$ . Then the system of equations becomes

$$\left\{ \begin{array}{ll} \frac{\partial \Gamma_{lj}}{\partial x_j} = F_l & \text{in } \Omega \\ -n_j \Gamma_{lj} = G_l + \frac{\partial R_m}{\partial u_l} \mu_m & \text{on } \partial\Omega \\ 0 = R_m & \text{on } \partial\Omega \end{array} \right.$$

#### *System for Two Variables in the General Form*

The following example of a PDE in the general form is a stationary system for  $N = 2$  solution components in  $n = 2$  space dimensions with  $M = 2$  constraints:

$$\left\{ \begin{array}{ll} \nabla \cdot \Gamma_1 = F_1 & \text{in } \Omega \\ \nabla \cdot \Gamma_2 = F_2 & \text{in } \Omega \end{array} \right.$$

with the generalized Neumann boundary conditions

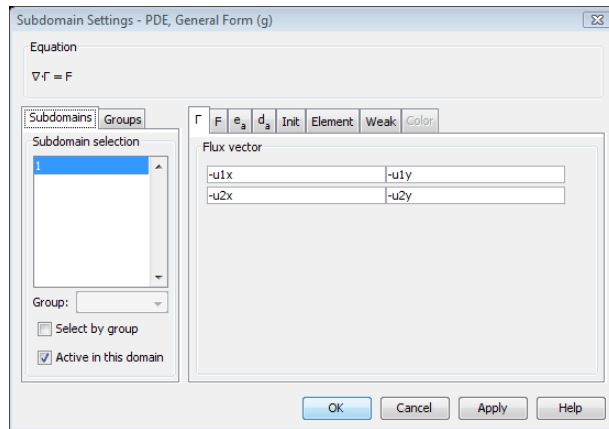
$$\left\{ \begin{array}{l} -\mathbf{n} \cdot \Gamma_1 = G_1 + \frac{\partial R_1}{\partial u_1} \mu_1 + \frac{\partial R_2}{\partial u_1} \mu_2 \quad \text{on } \partial\Omega \\ -\mathbf{n} \cdot \Gamma_2 = G_2 + \frac{\partial R_1}{\partial u_2} \mu_1 + \frac{\partial R_2}{\partial u_2} \mu_2 \quad \text{on } \partial\Omega \end{array} \right.$$

and the Dirichlet boundary conditions

$$\left\{ \begin{array}{l} 0 = R_1 \quad \text{on } \partial\Omega \\ 0 = R_2 \quad \text{on } \partial\Omega \end{array} \right.$$

### Specifying PDE Coefficients on Subdomains

The following image shows the **Subdomain Settings** dialog box for a general form, 2-variable stationary 2D problem:



### Specifying and Interpreting Boundary Conditions

The **Boundary Settings** dialog box is similar to the one for the coefficient form PDEs but in general form you specify only the  $g$  and  $r$  coefficients. To illustrate the flexibility of the boundary conditions, consider four cases:

- 1 Let  $R_1 = R_2 = 0$ . Then the Dirichlet boundary conditions give  $0 = 0$ . In addition, the terms containing the Lagrange multipliers disappear from the Neumann boundary condition. Thus you have only the Neumann boundary conditions



$$\begin{cases} -\mathbf{n} \cdot \Gamma_1 = G_1 & \text{on } \partial\Omega \\ -\mathbf{n} \cdot \Gamma_2 = G_2 & \text{on } \partial\Omega \end{cases}$$

- 2** Let  $R_1 = r_1 - u_1$  and  $R_2 = r_2 - u_2$ . Then the Dirichlet conditions are the usual  $u_1 = r_1$  and  $u_2 = r_2$ . The Neumann boundary conditions become

$$\begin{cases} -\mathbf{n} \cdot \Gamma_1 = G_1 - \mu_1 & \text{on } \partial\Omega \\ -\mathbf{n} \cdot \Gamma_2 = G_2 - \mu_2 & \text{on } \partial\Omega \end{cases}$$

These equations impose no restrictions on  $u_1$  or  $u_2$ , because the Lagrange multipliers  $\mu_1$  and  $\mu_2$  always adjust so as to fulfill the Dirichlet conditions. In this case, you can therefore ignore the Neumann boundary conditions.

- 3** Let  $R_1 = r_1 - u_1$  and  $R_2 = 0$ . Then the Dirichlet conditions are

$$\begin{cases} 0 = r_1 - u_1 & \text{on } \partial\Omega \\ 0 = 0 & \text{on } \partial\Omega \end{cases}$$

and the Neumann conditions are

$$\begin{cases} -\mathbf{n} \cdot \Gamma_1 = G_1 - \mu_1 & \text{on } \partial\Omega \\ -\mathbf{n} \cdot \Gamma_2 = G_2 & \text{on } \partial\Omega \end{cases}$$

The first Neumann condition can be ignored because it imposes no restriction on  $u_1$  or  $u_2$ . You effectively have only the Dirichlet condition on  $u_1$  together with the second Neumann condition.

- 4** The same as Case 3 above but with the two PDEs interchanged ( $\Gamma_1$  and  $\Gamma_2$  as well as  $F_1$  and  $F_2$ ). Then the PDEs are

$$\begin{cases} \nabla \cdot \Gamma_2 = F_2 & \text{in } \Omega \\ \nabla \cdot \Gamma_1 = F_1 & \text{in } \Omega \end{cases}$$

The Dirichlet condition is similar to that in Case 3:  $u_1 = r_1$ . The Neumann conditions then become

$$\begin{cases} -\mathbf{n} \cdot \Gamma_2 = G_2 - \mu_1 & \text{on } \partial\Omega \\ -\mathbf{n} \cdot \Gamma_1 = G_1 & \text{on } \partial\Omega \end{cases}$$

Effectively, you have only the Neumann condition  $-\mathbf{n} \cdot \Gamma_1 = G_1$ . In comparison with Case 3 above, the PDEs and the Dirichlet conditions are identical, while the Neumann conditions are different. Thus, when mixing Dirichlet and Neumann conditions, the ordering of the equations and the dependent variables are important. However, the ordering of the Dirichlet conditions does not matter since the different Lagrange multipliers are for all practical purposes indistinguishable from each other.

- 5 Finally, let  $R_1 = u_2 - u_1$  and  $R_2 = 0$ . Then the Dirichlet conditions are

$$\begin{cases} 0 = u_2 - u_1 & \text{on } \partial\Omega \\ 0 = 0 & \text{on } \partial\Omega \end{cases}$$

and the Neumann conditions are

$$\begin{cases} -\mathbf{n} \cdot \Gamma_1 = G_1 - \mu_1 & \text{on } \partial\Omega \\ -\mathbf{n} \cdot \Gamma_2 = G_2 + \mu_1 & \text{on } \partial\Omega \end{cases}$$

Note that the same Lagrange multiplier now appears in both Neumann conditions, which can have different definitions of  $\Gamma$  and  $G$ . Therefore, contrary to Cases 2 and 3 above, the Neumann conditions cannot be ignored. Instead, adding the two conditions, it becomes apparent that the solution and flux on the boundary must fulfill

$$\begin{cases} 0 = u_2 - u_1 & \text{on } \partial\Omega \\ -\mathbf{n} \cdot \Gamma_2 - \mathbf{n} \cdot \Gamma_1 = G_1 + G_2 & \text{on } \partial\Omega \end{cases}$$

In these examples, the values of the Lagrange multipliers do not matter. However, they often have a physical significance. In structural mechanics, the term

$$\left(\frac{\partial R}{\partial u}\right)^T \mu$$

in the Neumann condition is the reaction force necessary to satisfy the kinematic constraints described by the Dirichlet conditions. This term has a special form because of the variational principles that give rise to it (see “Variational Principles” on page 370).

### COEFFICIENT FORM VS. GENERAL FORM

The following substitutions show that the coefficient form and the general form are equivalent:

$$\Gamma = -c\nabla u - \alpha u + \gamma, \quad F = f - \beta\nabla u - \alpha u, \quad G = g - qu, \quad R = r - hu$$

---

**Note:** If  $r$  or  $h$  depend on  $u$ , there is a difference in the Neumann boundary condition because  $\partial R/\partial u$  need not equal  $-h$ .

---

This duality lets you choose the representation that best suits a particular PDE. You can always convert a problem to a more general form:

- From the coefficient form to the general form
- From the coefficient or the general form to the weak form

Make this conversion by selecting another equation system form in the **Model Settings** dialog box.

COMSOL Multiphysics converts the coefficient form PDE to a general form PDE according to

$$\begin{aligned}\Gamma_{lj} &= -c_{lkji} \frac{\partial u_k}{\partial x_i} - \alpha_{lkj} u_k + \gamma_{lj} \\ F_l &= f_l - \beta_{lki} \frac{\partial u_k}{\partial x_i} - \alpha_{lk} u_k \\ G_l &= g_l - q_{lk} u_k \\ R_m &= r_m - h_{ml} u_l\end{aligned}$$

using a notation where there is an implicit summation over the  $k$  (or  $l$ ) and  $i$  indices in each product. The conversion only applies to the PDE coefficients and the boundary coefficients. Other settings on boundaries and subdomains such as initial values, shape functions, and weak form contributions remain unchanged.

## Solving Time-Dependent Problems

To get the equation for a time-dependent PDE in COMSOL Multiphysics, add terms containing time derivatives to the left-hand side of the stationary equation. The time derivatives must appear linearly, and the Dirichlet conditions must be linear. A time-dependent problem in the coefficient form reads

$$\left\{ \begin{array}{ll} e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} + \nabla \cdot (-c \nabla u - \alpha u + \gamma) + \beta \cdot \nabla u + a u = f & \text{in } \Omega \\ \mathbf{n} \cdot (c \nabla u + \alpha u - \gamma) + q u = g - h^T \mu & \text{on } \partial\Omega \\ h u = r & \text{on } \partial\Omega \end{array} \right.$$

If  $u$  is a vector of dependent variables then the *mass coefficient*  $e_a$  is a matrix. All coefficients can depend on time. The name *mass matrix* or mass coefficient stems from the fact that in many physics applications  $e_a$  contains the mass density. The  $d_a$  coefficient represents damping for wave-type equations. However, if  $e_a = 0$ , then  $d_a$  is often called the mass coefficient. The default settings are  $e_a = 0$  and  $d_a = 1$ , representing a time-dependent PDE such as the heat equation. Using  $e_a = 1$  and  $d_a = 0$  represents an undamped wave equation.

For a stationary model, COMSOL Multiphysics ignores any values or expressions that you enter in the edit field for the  $d_a$  and  $e_a$  coefficient. To activate the  $d_a$  and  $e_a$  coefficients and convert the model into a time-dependent model, select a time-dependent solver in the **Solver Parameters** dialog box.

The time-dependent problem in the general form is

$$\left\{ \begin{array}{ll} e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} + \nabla \cdot \Gamma = F & \text{in } \Omega \\ -\mathbf{n} \cdot \Gamma = G + \left( \frac{\partial R}{\partial u} \right)^T \mu & \text{on } \partial\Omega \\ 0 = R & \text{on } \partial\Omega \end{array} \right.$$

The flux vector  $\Gamma$  and the scalar coefficients  $F$ ,  $G$ , and  $R$  can be functions of the spatial coordinates, the solution  $u$ , and the space and time derivatives of  $u$ . The superscript “ $T$ ” in the Neumann boundary condition denotes the transpose. The variable  $\mu$  is the Lagrange multiplier.

For the weak form, the time-dependent problem (in 2D) looks like

$$\begin{aligned} \int_{\Omega} W^{(2t)} dA + \int_B W^{(1t)} ds + \sum_P W^{(0t)} &= \int_{\Omega} W^{(2)} dA + \int_B W^{(1)} ds + \sum_P W^{(0)} + \\ &+ \int_{\Omega} v_l \frac{\partial R_m^{(2)}}{\partial u_l} \mu_m^{(2)} dA + \int_B v_l \frac{\partial R_m^{(1)}}{\partial u_l} \mu_m^{(1)} ds + \sum_P v_l \frac{\partial R_m^{(0)}}{\partial u_l} \mu_m^{(0)} \\ 0 &= R^{(2)} \quad \text{on } \Omega \\ 0 &= R^{(1)} \quad \text{on } B \\ 0 &= R^{(0)} \quad \text{on } P \end{aligned}$$

where the integrands  $W^{(it)}$  depend bilinearly on the test functions  $v_l$  and on the first and second time derivatives of the dependent variables and their space derivatives. For example,

$$W^{(2t)} = v_1 \frac{\partial u_2}{\partial t} - \frac{\partial v_1}{\partial x} \cos(3u_1) \frac{\partial u_1}{\partial t} + v_2 \frac{\partial^2 u_2}{\partial t \partial y}$$

#### Time-Dependent Systems

For time-dependent systems of PDEs the  $d_a$  and  $e_a$  coefficients are matrices. For example, for the system

$$\begin{aligned} e_{11} \frac{\partial^2 u}{\partial t^2} + e_{12} \frac{\partial^2 u_2}{\partial t^2} + d_{11} \frac{\partial u_1}{\partial t} + d_{12} \frac{\partial u_2}{\partial t} + \nabla \cdot (-c_{11} \nabla u_1 - c_{12} \nabla u_2) + a_{11} u_1 + a_{12} u_2 &= f_1 \\ e_{21} \frac{\partial^2 u}{\partial t^2} + e_{22} \frac{\partial^2 u_2}{\partial t^2} + d_{21} \frac{\partial u_1}{\partial t} + d_{22} \frac{\partial u_2}{\partial t} + \nabla \cdot (-c_{21} \nabla u_1 - c_{22} \nabla u_2) + a_{21} u_1 + a_{22} u_2 &= f_2 \end{aligned}$$

the coefficient matrices are

$$\begin{aligned} e_a &= \begin{bmatrix} e_{11} & e_{12} \\ e_{21} & e_{22} \end{bmatrix} & d_a &= \begin{bmatrix} d_{11} & d_{12} \\ d_{21} & d_{22} \end{bmatrix} \\ c &= \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} & a &= \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \end{aligned}$$

(where each element  $c_{ij}$  can be an  $n$ -by- $n$  matrix). Many interesting problems have a singular  $d_a$  matrix (with  $e_a = 0$ ), or a singular nonzero  $e_a$  matrix. Such problems are

called *differential-algebraic equation* (DAE) systems. The COMSOL Multiphysics solver for time-dependent problems handles DAEs.

The models “Example—Resistive Heating” on page 395 and “Thermal Effects in Electronic Conductors” on page 36 in the *COMSOL Multiphysics Quick Start and Quick Reference* are both DAE systems.

### *Solving Eigenvalue Problems*

---

COMSOL Multiphysics handles scalar eigenvalue problems for all PDE forms. These eigenvalue problems are related to time-dependent problems via the correspondence  $\partial/\partial t \leftrightarrow -\lambda$ , linking the time derivative to the eigenvalue  $\lambda$  (with the default eigenvalue name `lambda`). An eigenvalue problem in the coefficient form reads

$$\begin{cases} (\lambda - \lambda_0)^2 e_a u - (\lambda - \lambda_0) d_a u + \nabla \cdot (-c \nabla u - \alpha u) + \beta \cdot \nabla u + a u = f & \text{in } \Omega \\ \mathbf{n} \cdot (c \nabla u + \alpha u) + q u = -h^T \mu & \text{on } \partial\Omega \\ h u = r & \text{on } \partial\Omega \end{cases}$$

where  $\lambda_0$  is the linearization point for the eigenvalue. Note that the source terms are ignored if the solution form is coefficient form. If the general or weak solution forms are used, the source terms are not ignored if they depend on the solution components. If the coefficients depend on  $u$  or the eigenvalue  $\lambda$ , COMSOL Multiphysics performs a linearization of the problem about the linearization point  $u = u_0, \lambda = \lambda_0$ . The software also performs this linearization for eigenvalue problems in the general and weak forms, though in a slightly different way. See “The Linear or Linearized Model” on page 386 in the *COMSOL Multiphysics User’s Guide* for information about linearization.

To specify the linearization point  $u_0$ , use the settings in the **Values of variables not solved for and linearization point** area on the **Initial Value** page in the **Solver Manager** dialog box. To specify the linearization point  $\lambda_0$  use the **Eigenvalue linearization point** edit field in the **Solver Parameters** dialog box. You can also change the name of the variable that should be treated as the eigenvalue in the **Eigenvalue name** edit field.

## Interpreting PDE Coefficients

The COMSOL PDE formulations can model a variety of problems, but note that this documentation uses coefficient names that fall within the realm of continuum mechanics and mass transfer. For the coefficient form:

- $e_a$  is the *mass coefficient*.
- $d_a$  is a *damping coefficient* or a *mass coefficient*.
- $c$  is the *diffusion coefficient*.
- $\alpha$  is the *conservative flux convection coefficient*.
- $\beta$  is the *convection coefficient*.
- $a$  is the *absorption coefficient*.
- $\gamma$  is the *conservative flux source term*.
- $f$  is the *source term*.

$$e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} - \nabla \cdot (c \nabla u + \alpha u - \gamma) + \beta \cdot \nabla u + a u = f$$

There are many interesting PDE problems to which this interpretation does not apply. For instance, a time-harmonic PDE such as the Helmholtz equation represents a time-dependent phenomenon transformed into the frequency domain.

For the Neumann boundary condition of the coefficient form

$$\mathbf{n} \cdot (c \nabla u + \alpha u - \gamma) + q u = g - h^T \mu$$

- $q$  is the *boundary absorption coefficient*.
- $g$  is the *boundary source term*.

## Classical PDEs

Many classical PDEs are instances of the coefficient form. All the classical PDEs in this section have their own application modes. To find them, open the **Model Navigator**. In the **Application Modes** tree, within the **COMSOL Multiphysics>PDE Modes** section, find the **Classical PDEs** folder. The table below shows the available classical PDEs using two notations: the compact notation of vector analysis (used in this documentation) and an expanded mathematical notation.

TABLE 9-2: CLASSICAL PDES IN COMPACT AND STANDARD NOTATION

EQUATION	COMPACT NOTATION	STANDARD NOTATION (2D)
Laplace's equation	$-\nabla \cdot (\nabla u) = 0$	$-\frac{\partial}{\partial x} \frac{\partial u}{\partial x} - \frac{\partial}{\partial y} \frac{\partial u}{\partial y} = 0$
Poisson's equation	$-\nabla \cdot (c \nabla u) = f$	$-\frac{\partial}{\partial x} \left( c \frac{\partial u}{\partial x} \right) - \frac{\partial}{\partial y} \left( c \frac{\partial u}{\partial y} \right) = f$
Helmholtz equation	$-\nabla \cdot (c \nabla u) + au = f$	$-\frac{\partial}{\partial x} \left( c \frac{\partial u}{\partial x} \right) - \frac{\partial}{\partial y} \left( c \frac{\partial u}{\partial y} \right) + au = f$
Heat equation	$d_a \frac{\partial u}{\partial t} - \nabla \cdot (c \nabla u) = f$	$d_a \frac{\partial u}{\partial t} - \frac{\partial}{\partial x} \left( c \frac{\partial u}{\partial x} \right) - \frac{\partial}{\partial y} \left( c \frac{\partial u}{\partial y} \right) = f$
Wave equation	$e_a \frac{\partial^2 u}{\partial t^2} - \nabla \cdot (c \nabla u) = f$	$e_a \frac{\partial^2 u}{\partial t^2} - \frac{\partial}{\partial x} \left( c \frac{\partial u}{\partial x} \right) - \frac{\partial}{\partial y} \left( c \frac{\partial u}{\partial y} \right) = f$
Schrödinger equation	$-\nabla \cdot (c \nabla u) + au = \lambda u$	$-\frac{\partial}{\partial x} \left( c \frac{\partial u}{\partial x} \right) - \frac{\partial}{\partial y} \left( c \frac{\partial u}{\partial y} \right) + au = \lambda u$
Convection-diffusion equation	$d_a \frac{\partial u}{\partial t} - \nabla \cdot (c \nabla u) + \beta \cdot \nabla u = f$	$d_a \frac{\partial u}{\partial t} - \frac{\partial}{\partial x} \left( c \frac{\partial u}{\partial x} \right) - \frac{\partial}{\partial y} \left( c \frac{\partial u}{\partial y} \right) + \beta_x \frac{\partial u}{\partial x} + \beta_y \frac{\partial u}{\partial y} = f$

## Equation Variables

The equation variables denote certain terms in the equations in coefficient form. Equation variables are available only when you have selected the coefficient or general solution form. Use the **Solution form** setting on the **Advanced** page in the **Solver Parameters** dialog box to set the solution form to **Coefficient** or **General**. Avoid using equation variables if there are other alternatives because you cannot use these variables



when using the weak solution form. Often the application mode already contains variables with the corresponding quantities defined as application mode variables. The section “Using the Weak Solution Form” on page 352 explains how the weak solution form might be of advantage.

Recall the coefficient form system of equations

$$\begin{cases} e_{a\ l k} \frac{\partial^2 u_k}{\partial t^2} + d_{a\ l k} \frac{\partial u_k}{\partial t} + \nabla \cdot (-c_{l k} \nabla u_k - \alpha_{l k} u_k + \gamma_l) + \beta_{l k} \cdot \nabla u_k + a_{l k} u_k = f_l & \text{in } \Omega \\ \mathbf{n} \cdot (c_{l k} \nabla u_k + \alpha_{l k} u_k - \gamma_l) + q_{l k} u_k = g_l - h_{m l} u_m & \text{on } \partial\Omega \\ h_{m k} u_k = r_m & \text{on } \partial\Omega \end{cases}$$

where  $k$  and  $l$  range from 1 to  $N$ , and  $m$  ranges from 1 to  $M$ , where  $M \leq N$ . Let  $y$  be the name of space coordinate number  $j$ . The notation in the following table uses the summation convention; that is, there is an implicit sum over all pairs of equal indices.

TABLE 9-3: EQUATION VARIABLES ON SUBDOMAINS

VARIABLE	MEANING	DESCRIPTION
dau $l$	$d_{a\ l k} u_k$	Mass coefficient multiplied with the solution
cu $l y$	$c_{l k j i} \partial u_k / \partial x_i$	Component of the diffusive flux
alu $l y$	$\alpha_{l k j} u_k$	Component of the conservative convective flux
ga $l y$	$\gamma_{l j}$	Component of the conservative flux source
beu $l$	$\beta_{l k j} \partial u_k / \partial x_j$	Convection term
au $l$	$a_{l k} u_k$	Absorption term
f $l$	$f_l$	Source term

TABLE 9-4: EQUATION VARIABLES ON BOUNDARIES

VARIABLE	MEANING	DESCRIPTION
qu $l$	$q_{l k} u_k$	Boundary absorption term
g $l$	$g_l$	Boundary source term

When there is only one dim variable, omit  $l$  from the equation variable names.

---

**Note:** In equation definitions, use the equation variables with caution to avoid circular definitions.

---

### *Boundary-Coupled Equation Variables*

---

The boundary-coupled equation variables extrapolate the values of the equation variables to boundaries. The variables are available only when you have selected the coefficient or general solution form. To specify one of those solution forms, go to the **Solve** menu and open the **Solver Parameters** dialog box; click the **Advanced** tab and in the **Solution form** list select **Coefficient** or **General**. Avoid using equation variables if there are other alternatives because you cannot use these variables when using the weak solution form, which is the one in use for most cases when using the automatic solution form selection. Often the corresponding variables are already defined by the application mode as application mode variables. The section “Weak Form vs. Strong Forms” on page 353 explains why the weak solution form can be of advantage.

Recall the coefficient form system of equations

$$\left\{ \begin{array}{ll} e_{a\ l k} \frac{\partial^2 u_k}{\partial t^2} + d_{a\ l k} \frac{\partial u_k}{\partial t} + \nabla \cdot (-c_{l k} \nabla u_k - \alpha_{l k} u_k + \gamma_l) + \beta_{l k} \cdot \nabla u_k + a_{l k} u_k = f_l & \text{in } \Omega \\ \mathbf{n} \cdot (c_{l k} \nabla u_k + \alpha_{l k} u_k - \gamma_l) + q_{l k} u_k = g_l - h_{m l} u_m & \text{on } \partial\Omega \\ h_{m k} u_k = r_m & \text{on } \partial\Omega \end{array} \right.$$

The first type of boundary-coupled equation variables consists of equation variables restricted to the boundaries. Let  $l$  be an integer and  $y$  the name of space coordinate number  $j$ . The variables in the following table are defined on the boundaries. The upper and lower subdomains are the subdomains in the upward and downward directions (see the definition of normal vectors in “Direction of the Normal Component on Interior Boundaries” on page 172 of the *COMSOL Multiphysics*

*User's Guide*). The table notation uses the summation convention, that is, there is an implicit summation over each pair of equal indices.

TABLE 9-5: BOUNDARY-COUPLED EQUATION VARIABLES

VARIABLE	MEANING
uculy	Value of $c_{lkji} \partial u_k / \partial x_i$ taken from the upper adjacent subdomain. 0 if there is no upper adjacent subdomain.
dculy	Value of $c_{lkji} \partial u_k / \partial x_i$ taken from the lower adjacent subdomain. 0 if there is no lower adjacent subdomain.
culy	If there are two adjacent subdomains, $(uculy+dculy)/2$ . If there is one adjacent subdomain, the value $c_{lkji} \partial u_k / \partial x_i$ taken from this subdomain. Otherwise undefined.
ualuly	Value of $\alpha_{lkj} u_k$ taken from the upper adjacent subdomain (0 if such a subdomain does not exist)
daluly	Value of $\alpha_{lkj} u_k$ taken from the lower adjacent subdomain (0 if such a subdomain does not exist)
aluly	If there are two adjacent subdomains, $(ualuly+daluly)/2$ . If there is one adjacent subdomain, the value $\alpha_{lkj} u_k$ taken from this subdomain. Otherwise undefined.
ugaly	Value of $\gamma_{lj}$ taken from the upper adjacent subdomain (0 if such does not exist)
dgalyl	Value of $\gamma_{lj}$ taken from the lower adjacent subdomain (0 if such a subdomain does not exist)
galyl	If there are two adjacent subdomains, $(ugaly+dgalyl)/2$ . If there is one adjacent subdomain, the value $\gamma_{lj}$ taken from this subdomain. Otherwise undefined.
ubeul	Value of $\beta_{lkj} \partial u_k / \partial x_j$ taken from the upper adjacent subdomain (0 if such a subdomain does not exist)
dbeul	Value of $\beta_{lkj} \partial u_k / \partial x_j$ taken from the lower adjacent subdomain (0 if such a subdomain does not exist)
beul	If there are two adjacent subdomains, $(ubeul+dbeul)/2$ . If there is one adjacent subdomain, the value taken from this subdomain. Otherwise undefined.
uau1	Value of $a_{lk} u_k$ taken from the upper adjacent subdomain (0 if such a subdomain does not exist)
dau1	Value of $a_{lk} u_k$ taken from the lower adjacent subdomain (0 if such a subdomain does not exist)
au1	If there are two adjacent subdomains, $(uau1+dau1)/2$ . If there is one adjacent subdomain, the value $a_{lk}$ taken from this subdomain. Otherwise undefined.

TABLE 9-5: BOUNDARY-COUPLED EQUATION VARIABLES

VARIABLE	MEANING
$uf_l$	Value of $f_l$ taken from the upper adjacent subdomain (0 if such a subdomain does not exist)
$df_l$	Value of $f_l$ taken from the lower adjacent subdomain (0 if such a subdomain does not exist)
$f_l$	If there are two adjacent subdomains, $(uf_l+df_l)/2$ . If there is one adjacent subdomain, the value $f_l$ taken from this subdomain. Otherwise undefined.

When there is only one dependent variable, omit  $l$  from these variable names.

The second type of boundary-coupled equation variables involves the normal vector of the boundary. Some of these variables represent the jump in the terms in the normal component of the flux vector. Let  $n_j$  denote the components of the outward normal vector, seen from the upper or lower adjacent subdomain. The table notations use the summation convention.

TABLE 9-6: BOUNDARY COUPLED EQUATION VARIABLES INVOLVING NORMAL TO BOUNDARY

VARIABLE	MEANING
$uncu_l$	Value of $n_j c_{lkji} \partial u_k / \partial x_i$ taken from the upper adjacent subdomain. 0 if there is no upper adjacent subdomain.
$dncu_l$	Value of $n_j c_{lkji} \partial u_k / \partial x_i$ taken from the lower adjacent subdomain. 0 if there is no lower adjacent subdomain.
$ncu_l$	$uncu_l+dncu_l$ . Since the normal vectors used in $uncu_l$ and $dncu_l$ are opposite, $ncu_l$ is a jump in normal diffusive flux.
$una_l u_l$	Value of $n_j \alpha_{lkj} u_k$ taken from the upper adjacent subdomain (0 if such a subdomain does not exist)
$dna_l u_l$	Value of $n_j \alpha_{lkj} u_k$ taken from the lower adjacent subdomain (0 if such a subdomain does not exist)
$na_l u_l$	$una_l u_l+dna_l u_l$
$unga_l$	Value of $n_j \gamma_{lj}$ taken from the upper adjacent subdomain (0 if such a subdomain does not exist)
$dnga_l$	Value of $n_j \gamma_{lj}$ taken from the lower adjacent subdomain (0 if such a subdomain does not exist)
$nga_l$	$unga_l+dnga_l$

When there is only one dependent variable, omit  $l$  from the above variable names.

---

**Note:** In equation definitions, use boundary-coupled equation variables with caution to avoid circular definitions.

---

### *Ideal and Non-Ideal Constraints*

---

Constraints formulated through the coefficients  $h$  and  $r$  in the PDE, Coefficient Form application mode and through the coefficient  $R$  in the PDE, General Form application mode, give rise to constraints called *ideal*. An ideal constraint dictates exactly how the the flux conditions (or Neumann boundary conditions) are influenced by the constraint force. For the coefficient form, the flux condition is

$$\mathbf{n} \cdot (c \nabla u + \alpha u) + qu = g - h^T \mu,$$

and for the general form, the flux condition is

$$-\mathbf{n} \cdot \Gamma = G + \left( \frac{\partial R}{\partial u} \right)^T \mu.$$

The last term on the right-hand side in both expressions is the ideal constraint force. Thus, with ideal constraints you cannot enforce a flux condition independently of the constraints. In mathematics as well as in multiphysics modeling it is often necessary to enforce Neumann conditions and Dirichlet conditions more freely than what is possible through ideal constraints. As an example, consider again the general form example in Case 3 on page 261 and assume that you want to enforce the boundary conditions

$$\begin{cases} 0 = r_1 - u_1 & \text{on } \partial\Omega \\ -\mathbf{n} \cdot \Gamma_2 = G_2 & \text{on } \partial\Omega \end{cases}.$$

If  $r_1 = r_1(u_2)$ , the first condition is fulfilled but not the second. This is because the ideal constraint force is not zero:

$$-\mathbf{n} \cdot \Gamma_2 = G_2 + \frac{\partial R_1}{\partial u_2} \mu_1 + \frac{\partial R_2}{\partial u_2} \mu_2 = G_2 + \frac{\partial r_1}{\partial u_2} \mu_1 \neq G_2.$$

To remedy this limitation with ideal constraints you can use *non-ideal constraints*, by which you can specify the constraint force independently of the constraints. The term non-ideal refers to the fact that the constraint force differs from  $-h^T \mu$  for coefficient form and from  $\left(\frac{\partial R}{\partial u}\right)^T \mu$  for general form. In multiphysics modeling, non-ideal constraints are, for example, necessary for the following boundary conditions:

- Normal-direction constraints on a moving mesh, where the mesh motion is part of the problem. These conditions are of the type  $\mathbf{n} \cdot \mathbf{u} - r = 0$  where  $\mathbf{n} = \mathbf{n}(\mathbf{x})$  is the boundary normal,  $\mathbf{u}$  is a vector field (displacements or velocity), and  $\mathbf{x}$  is the mesh coordinate vector. Ideal constraints give constraint forces not only on the equations for  $\mathbf{u}$  but also on the equations for  $\mathbf{x}$ , which typically are not wanted.
- Wall boundary conditions for turbulent fluid flow. For the  $k$ - $\varepsilon$  turbulence model this condition is of the type  $k - r(\varepsilon)$ ,  $-n \cdot \nabla \varepsilon = 0$ , where  $r$  is a given function. Ideal constraints for the first relation imply that the second relation cannot hold.

Non-ideal constraints can be enforced in COMSOL Multiphysics both in a pointwise and in a weak sense. For descriptions of how to use pointwise and weak non-ideal constraints see “Constraint Forces” on page 276 and “Specifying Weak Constraints” on page 359, respectively.

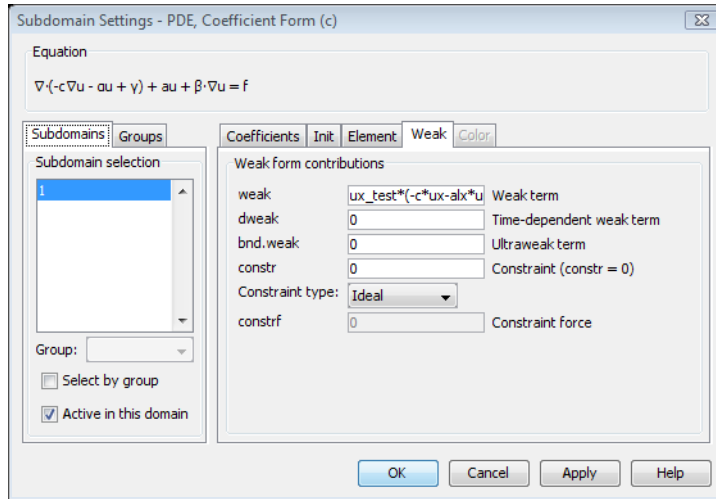
### *Variables for PDEs in Weak Form*

---

When you use the weak solution form (the default solution form with the automatic solution form setting), the shape function variables (and the boundary-coupled shape variables) are the only automatically generated field variables available as defaults. The shape function variables are available in the same way as described earlier for the coefficient form and general solution form. User-defined expression variables and coupling variables are also available.

In addition to shape function variables, meta variables for test functions, test-function derivatives, and time derivatives are available in the **weak** and **dweak** edit fields on the **Weak** page in the **Subdomain Settings**, **Boundary Settings**, **Edge Settings**, and **Point Settings** dialog boxes. The test function meta variables follow a naming convention similar to that used for variables; see “Variable Naming Conventions” on page 169 of the *COMSOL Multiphysics User’s Guide*.

To add weak terms, click the **Weak** tab in the **Point Settings**, **Edge Settings**, **Boundary Settings**, and **Subdomain Settings** dialog boxes. For the PDE, Coefficient Form application mode, the following dialog box is available for subdomains.



In the edit fields on the **Weak** page you can type a series of weak terms. The test functions always occur linearly in the weak terms. You can use these weak terms either alone to define the problem, as when using the weak form application modes, or in addition to the other coefficients in the coefficient form and general form application modes.

The following table shows the available test-function meta variables for a 2-variable weak form PDE. Let  $u$  be the name of the dependent variable and  $x$ ,  $y$ , and  $z$  be the names of the independent variables.

	1D	2D	3D
<b>WEAK FORM META VARIABLE</b>	$u\_test, ux\_test$	$u\_test, ux\_test, uy\_test$	$u\_test, ux\_test, uy\_test, uz\_test$

The corresponding 2nd-order space derivatives are also available.

There is also a test operator with the same effect as the meta variables, so an alternative syntax to  $u1\_test$ , for example, is  $test(u1)$ . See “Using Special Operators” on page 163 of the *COMSOL Multiphysics User’s Guide* for more information about the test operator and other operators.

## CONSTRAINT FORCES

On the **Weak** page you can also specify which constraint type to use. The default constraint type is *ideal*, which means that the PDE, Coefficient Form and PDE, General Form application modes derive the constraint force directly from the constraint specified in the **constr** field. Constraints specified using the **h** and **r** edit fields on the **Coefficients** page are always ideal and the **Constraint type** setting has no effect on them.

If you choose to use the default *non-ideal* constraints, the constraint force from a constraint written as **constr** =  $R = 0$  on  $\partial\Omega$ , changes according to

$$\left(\frac{\partial R}{\partial u}\right)^T \mu \rightarrow -I_{\partial\Omega}\mu$$

where  $I_{\partial\Omega}$  is an identity operator for points on the boundary  $\partial\Omega$ . If, for example, a two-variable problem for variables  $u_1$  and  $u_2$  specifies the constraints (using the **constr** field)

$$\begin{cases} 0 = R_1(u_1, u_2) & \text{on } \partial\Omega \\ 0 = R_2(u_1, u_2) & \text{on } \partial\Omega \end{cases},$$

then the default non-ideal constraint force is  $-\mu_1$  acting on  $u_1$  and  $-\mu_2$  acting on  $u_2$ . The corresponding ideal constraint would give reaction forces

$$\left(\frac{\partial R_1}{\partial u_1}\right)^T \mu_1 + \left(\frac{\partial R_2}{\partial u_1}\right)^T \mu_2$$

acting on  $u_1$  and

$$\left(\frac{\partial R_1}{\partial u_2}\right)^T \mu_1 + \left(\frac{\partial R_2}{\partial u_2}\right)^T \mu_2$$

acting on  $u_2$ .

If you want a constraint force different from any of these two choices, you can specify a separate expression for it. Select **User defined** from the **Constraint type** list and enter the expression in the **constf** edit field. In this edit field you specify the constraint force Jacobian using the test operator. For example, the expression corresponding to the default non-ideal constraint in the example above is



$$\begin{cases} -\text{test}(u1) \\ -\text{test}(u2) \end{cases}$$

**Note:** The ideal constraint type normally generates a symmetric constraint Jacobian matrix and constraint force Jacobian matrix coupling for the discretized equations, while non-ideal constraints generate an unsymmetric coupling. See also “The Discretized Linearized Model” on page 388 of the *COMSOL Multiphysics User’s Guide*.

### *Application Mode Variables*

The following list shows the application mode variables that are available for postprocessing in addition to the other variables for PDEs:

NAME	TYPE	DESCRIPTION	EXPRESSION
$u_i$	S, B	The solution variable (dependent variable)	$u_i$
$u_{ixj}$	S, B	The derivative of the solution variable $u_i$ with respect to the space coordinate $x_j$ , for example, $u_y$	$\frac{\partial u_i}{\partial x_j}$
$u_{ixjxk}$	S, B	The second derivative of the solution variable $u_i$ with respect to the space coordinates $x_j$ and $x_k$ , for example, $u_{xx}$ , $u_{xy}$ .	$\frac{\partial^2 u_i}{\partial x_j \partial x_k}$
$u_{it}$	S, B	The derivative of the solution variable $u_i$ with respect to time	$\frac{\partial u_i}{\partial t}$
$u_{itt}$	S, B	The second derivative of the solution variable $u_i$ with respect to time	$\frac{\partial^2 u_i}{\partial t^2}$
$u_{ixjt}$	S, B	The mixed derivative of the solution variable $u_i$ with respect to time and the space coordinate $x_j$ .	$\frac{\partial^2 u_i}{\partial x_j \partial t}$

NAME	TYPE	DESCRIPTION	EXPRESSION
absu <sub>i</sub> x <sub>i</sub>	S, B	Gradient of solution	$ \nabla u , \frac{\partial u_i}{\partial x_i}$
abscu <sub>i</sub> x	S	Value of the c*grad( $u_i$ ) vector (Coefficient Form only). Only available when solving using coefficient form as solution form.	$ c \nabla(u) $
absga <sub>i</sub> x	S	Value of the Gamma vector (General Form only). Only available when solving using general form as solution form.	$ \Gamma $
cu <sub>i</sub> x	S, B	c*grad( $u_i$ ) vector components (Coefficient Form only). Only available when solving using coefficient form as solution form.	$c \frac{\partial u_i}{\partial x}$
ga1x <sub>i</sub>	S, B	Gamma vector components (General Form only). Only available when solving using general form as solution form.	$\gamma_x$
ncu <sub>i</sub>	B	Normal components of the c*grad( $u_i$ ) vector (Coefficient Form only). Only available when solving using coefficient form as solution form.	$\mathbf{n} \cdot c \nabla u$
nga <sub>i</sub>	B	Normal components of the Gamma vector (General Form only). Only available when solving using general form as solution form.	$\mathbf{n} \cdot \gamma$

**Note:** All application mode variables include a suffix indicating which application mode they belong to. Table 9-1 on page 246 lists the default suffix for the PDE modes. For example, the default suffix added to application mode variable names for a General Form application mode is `_g`. The dependent variables  $u_i$  and their space and time derivatives do not include a suffix.

## *PDE Coefficients and Boundary Conditions with Time Derivatives*

---

The  $d$  coefficients relate to 1st-order time derivatives of the solution  $u$ . The only directly available  $d$  coefficient is  $d_a$ , the coefficient in front of  $\partial u/\partial t$ ; the subscript  $a$  is used because  $d_a$  is similar to the  $a$  coefficient in the absorption term except that it multiplies  $\partial u/\partial t$  instead of  $u$ . In contrast, the coefficients  $d_c$ ,  $\mathbf{d}_\alpha$ , and  $\mathbf{d}_\beta$  for  $\partial u/\partial t$ —analogous to the coefficients  $c$ ,  $\alpha$ , and  $\beta$  for  $u$ , respectively—in the extended PDE formulation

$$-\nabla \cdot \left( d_c \nabla \frac{\partial u}{\partial t} + \mathbf{d}_\alpha \frac{\partial u}{\partial t} \right) + \mathbf{d}_\beta \cdot \nabla \frac{\partial u}{\partial t} + d_a \frac{\partial u}{\partial t} + \nabla \cdot (-c \nabla u + \dots)$$

are not directly available in the general or coefficient form PDE models. Instead, you can enter them in the existing  $\gamma$  and  $f$  terms:

- In 1D, add  $-d_c * u_{xt} - d_{a1} * u_t$  to the  $\gamma$  term, and add  $-d_{be} * u_{xt}$  to the  $f$  term.
- In 2D, add  $-d_c * u_{xt} - d_{a11} * u_t$  to the first  $\gamma$  component, and add  $-d_c * u_{yt} - d_{a12} * u_t$  to the second  $\gamma$  component. Add  $-d_{be1} * u_{xt} - d_{be2} * u_{yt}$  to the  $f$  term.

Replace the variables  $d_c$ ,  $d_{a1}$ ,  $d_{be}$ ,  $d_{a11}$ ,  $d_{a12}$ ,  $d_{be1}$ , and  $d_{be2}$  with appropriate expressions.

You enter 2nd-order time derivative terms in an analogous manner.

To specify a boundary condition containing time-derivative terms as in

$$\mathbf{n} \cdot (c \nabla u + \dots) + e_q \frac{\partial^2 u}{\partial t^2} + d_q \frac{\partial u}{\partial t} + qu = g - h^T \mu,$$

add the terms  $-e_q * u_{tt} - d_q * u_t$  to the  $g$  term, and provide appropriate values or expressions for the coefficients  $e_q$  and  $d_q$  in, for instance, the **Options>Expressions>Boundary Expressions** dialog box.

# Using Equation Contributions on Interior Mesh Boundaries

In COMSOL Multiphysics it is possible to have equation contributions also from mesh element boundaries that does not lie on a geometry boundary. This makes additional finite element formulations possible like, for example, the discontinuous Galerkin method in space for some transport and flow problems, and the ultraweak variational formulation for wave problems driven by a single frequency.

---

**Note:** The Acoustics Module provides a predefined interface for modeling pressure acoustics using the ultraweak variational formulation.

---

You specify these interior mesh boundary equation contributions on the corresponding subdomains. It is also possible to define expression variables to use on the interior mesh boundaries. Useful applications typically use discontinuous elements and the up and down operators in expressions to access values on elements adjacent to the interior mesh boundaries.

## *Specifying Equation Contributions*

---

- 1 Open the **Subdomain Settings** dialog box from the **Physics** menu for any of the PDE Modes. For models with other application modes, choose **Equation System > Subdomain Settings**.
- 2 Select the subdomain where you want to enter an equation contribution on the interior mesh boundaries.
- 3 Click the **Weak** tab or the **bnd.weak** tab depending on the equation system form.
- 4 Enter your equation contributions in the **bnd.weak** edit field or the edit fields in the **bnd.weak** terms area.
- 5 Click the **Element** tab.
- 6 Enter the desired integration order for the ultraweak equation contributions in the **bnd.gporder** text field.
- 7 Click **OK**.

### *Specifying Interior Mesh Boundary Expressions.*

---

To specify expression variables on the interior mesh boundaries, select **Interior Mesh Boundary Expressions** from the **Expressions** submenu on the **Options** menu.

The model example “A Transport Problem” on page 179 of the *COMSOL Multiphysics Model Library* shows how to implement the discontinuous Galerkin method with upwinding stabilization using equation contributions on interior mesh boundaries.

# Examples—Equation-Based Modeling

The models in this section solve Poisson’s equation on the unit disk, first with a constant source term and then with a point source at the origin. In both cases, the exact solutions are known, making a quantitative assessment of the accuracy of the numerical results straightforward.

The models serve as introductions to equation-based modeling using PDEs in COMSOL Multiphysics. You find many more equation-based models in the chapter “Equation-Based Models” of the *COMSOL Multiphysics Model Library*.

## *Poisson’s Equation on the Unit Disk*

---

A classic PDE with a well-known behavior is Poisson’s equation

$$\begin{cases} -\nabla \cdot (\nabla u) = f & \Omega \\ u = 0 & \partial\Omega \end{cases}$$

on the unit disk  $\Omega$  with  $f = 1$ . The exact solution is

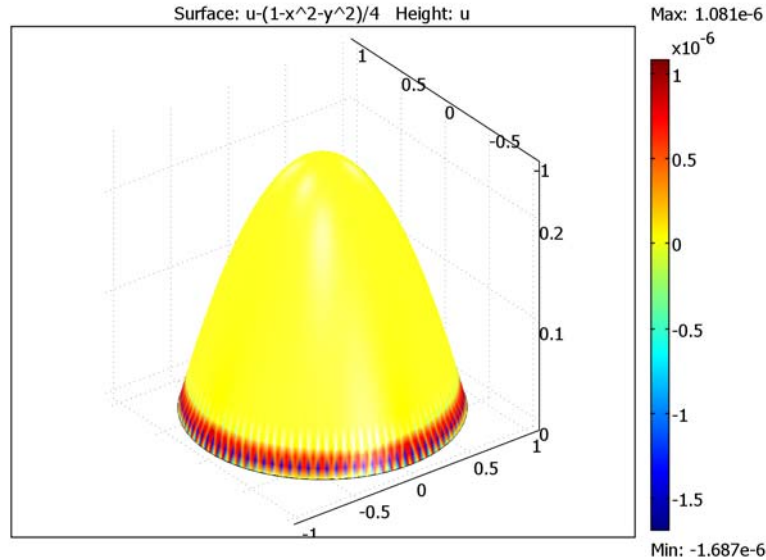
$$u(x, y) = \frac{1 - x^2 - y^2}{4},$$

which makes it possible to compare the COMSOL Multiphysics solution with the values of the exact solution at the node points on the mesh.

## Results

---

The plot below shows the error (the difference between the numeric solution in COMSOL Multiphysics and the exact, analytic solution):



The finite element solution is very good near the center of the domain, which is natural considering that the exact solution and the element shape functions are both 2nd-order polynomials. Close to the boundary, however, the error becomes much larger. Because the element shape at the boundary follows the geometry, the relationship between local and global coordinates is not linear. Therefore, while being 2nd-order polynomials in their own local coordinates, the shape functions are not strictly 2nd-order polynomials in  $x$  and  $y$ .

---

**Model Library path:** COMSOL\_Multiphysics/Benchmarks/poisson\_unit\_disk

---

## Modeling Using the Graphical User Interface

---

The following section explains how to derive the above plot using the graphical user interface.

## MODEL NAVIGATOR

- 1 In the **Model Navigator**, select **2D** from the **Space dimension** list.
- 2 In the list of application modes, open the **COMSOL Multiphysics>PDE Modes** folder and then the **Classical PDEs** folder.
- 3 Select **Poisson's Equation**.
- 4 Select **Lagrange - Quadratic** in the **Element** list (the default element type).
- 5 Click **OK**.

## OPTIONS AND SETTINGS

- 1 Go to the **Options** menu and choose **Axes/Grid Settings**.
- 2 Enter the axis limits from the following table.
- 3 Click the **Grid** tab.
- 4 Clear the **Auto** check box and enter the following values for the grid spacings and extra grid points:

AXIS		GRID	
x min	-2	x spacing	0.5
x max	2	Extra x	
y min	-1.5	y spacing	0.5
y max	1.5	Extra y	

## GEOMETRY MODELING

- 1 Find the Draw toolbar and click the **Ellipse/Circle (Centered)** button.
- 2 Using the right mouse button, draw a circle centered at (0, 0) with a radius of 1.

## PHYSICS SETTINGS

### *Boundary Conditions*

The default boundary condition in this mode is  $u = 0$ , so no changes are necessary.

### *Subdomain Settings*

By default  $f$  is set to 1. Again, no changes are necessary.

## MESH GENERATION

- 1 Click the **Mesh Mode** button on the Main toolbar to initialize and display the mesh.
- 2 Click the **Refine Mesh** button on the Main toolbar.



## COMPUTING THE SOLUTION

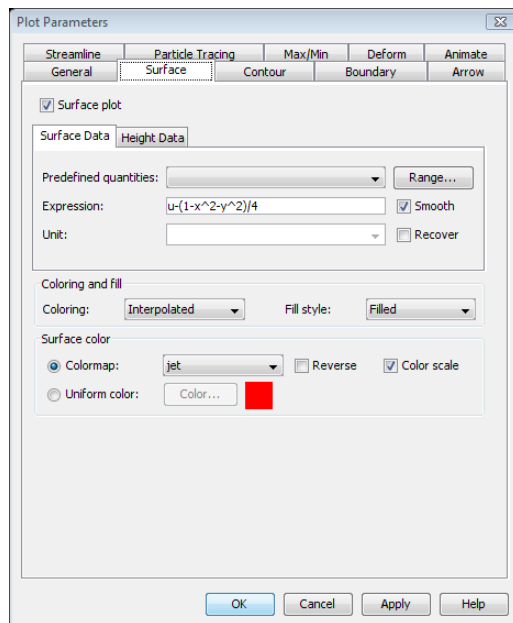
Click the **Solve** button on the Main toolbar.

## POSTPROCESSING AND VISUALIZATION

To view the solution as a 3D surface, click the **3D Surface Plot** button on the Plot toolbar.

Compare the FEM solution to the exact solution:

- 1 Click the **Plot Parameters** toolbar button.
- 2 Click the **Surface** tab.
- 3 On the **Surface Data** page, type  $u - (1 - x^2 - y^2)/4$  in the **Expression** edit field.



- 4 Click **OK**.

### *Implementing a Point Source*

Consider Poisson's equation on the unit circle with a point source at the origin. Its formal expression is:

$$\begin{cases} -\nabla \cdot (\nabla u) = \delta & \Omega \\ u = 0 & \partial\Omega \end{cases}$$

where  $\delta$  is the Dirac  $\delta$  distribution located at the origin. The exact solution is  $u = -\frac{1}{2\pi} \log(r)$ , which has a singularity at the origin.

The easiest way to describe a point source is with an extra weak term. Understanding why requires some basic knowledge of FEM theory and the weak formulation of PDEs. To obtain the weak formulation of the general Poisson equation (where  $f$  is any source term), multiply it with a test function  $u_{\text{test}}$  and integrate over the domain  $\Omega$ :

$$\begin{cases} \int_{\Omega} -\nabla \cdot (\nabla u) u_{\text{test}} = \int_{\Omega} f u_{\text{test}} \\ \int_{\partial\Omega} u u_{\text{test}} = 0 \end{cases}, \forall u_{\text{test}}$$

Integrate by parts and introduce the Lagrange multiplier  $\lambda$  to account for the constraint on the boundary. This results in the weak formulation:

$$\begin{cases} \int_{\Omega} \nabla u \cdot \nabla u_{\text{test}} + \int_{\partial\Omega} \lambda u_{\text{test}} = \int_{\Omega} f u_{\text{test}} \\ \int_{\partial\Omega} u u_{\text{test}} = 0 \end{cases}, \forall u_{\text{test}}$$

Most terms in these equations are sums of integrals over the domain,  $\Omega$ , or over the domain boundary,  $\partial\Omega$ . You can explicitly add extra weak terms to the equations. In addition to handling terms integrated over  $\Omega$  or  $\partial\Omega$ , COMSOL Multiphysics also handles contributions from integrals over edges and from single points. The software does not integrate contributions from points but instead collects them and adds them directly to the equations.

COMSOL Multiphysics discretizes the integral equations just given as long as the source term  $f$  is a function of the space coordinates, solution variables, and time. Unfortunately, you cannot express the Dirac  $\delta$  distribution as a function that COMSOL Multiphysics can integrate numerically. The solution lies in noting that by the definition of the Dirac  $\delta$  distribution, the following equality holds true:

$$\int_{\Omega} \delta u_{\text{test}} = u_{\text{test}}(0)$$

Therefore, setting  $f$  to zero and adding  $u_{\text{test}}$  as a weak term for a point at the origin provides the problem's correct weak formulation. The following model handles the resolution of the singularity at the origin using local mesh refinement.

---

**Model Library path:** COMSOL\_Multiphysics/Benchmarks/point\_source

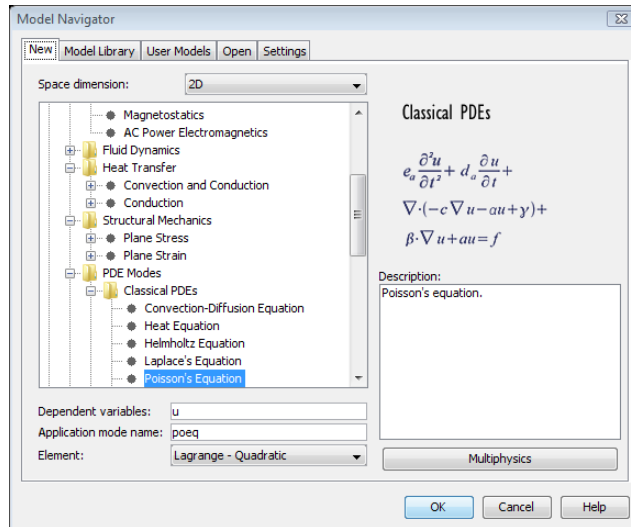
---

### *Modeling Using the Graphical User Interface*

---

#### MODEL NAVIGATOR

- 1 In the **Model Navigator**, select **2D** from the **Space dimension** list.
- 2 In the list of application modes, open the **COMSOL Multiphysics>PDE Modes** folder and then **Classical PDEs**.
- 3 Select **Poisson's Equation**. In the **Element** list be sure to select **Lagrange - Quadratic**.



- 4 Click **OK**.

## OPTIONS AND SETTINGS

- 1 Go to the **Options** menu and choose **Axes/Grid Settings**.
- 2 In the **Axes/Grid Settings** dialog box, enter the following axis limits:

AXIS	
x min	-2
x max	2
y min	-1.5
y max	1.5

- 3 Click the **Grid** tab.
- 4 Clear the **Auto** check box.
- 5 Enter the following values in the edit fields for the grid spacings; when finished, click **OK**.

GRID	
x spacing	0.5
Extra x	
y spacing	0.5
Extra y	

## GEOMETRY MODELING

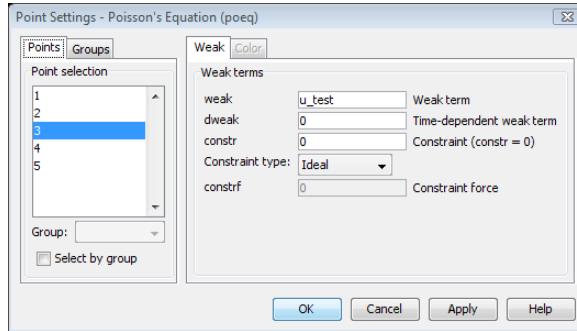
- 1 Click the **Ellipse/Circle (Centered)** button on the Draw toolbar.
- 2 Using the right mouse button, draw a circle centered at  $(0, 0)$  with a radius of 1.
- 3 Click the **Point** button on the Draw toolbar and click once at the origin.

## PHYSICS SETTINGS

### *Point Settings*

- 1 Go to the **Physics** menu and choose **Point Settings**.
- 2 In the **Point Settings** dialog box select Point 3.

3 Type `u_test` in the **weak** edit field.



4 Click **OK**.

### *Boundary Conditions*

The default boundary condition in this mode is  $u = 0$ , so you do not need to change anything.

### *Subdomain Settings*

- 1 From the **Physics** menu choose **Subdomain Settings**.
- 2 Select Subdomain 1.
- 3 Enter these PDE coefficients; when finished, click **OK**.

PROPERTY	VALUE
c	1
f	0

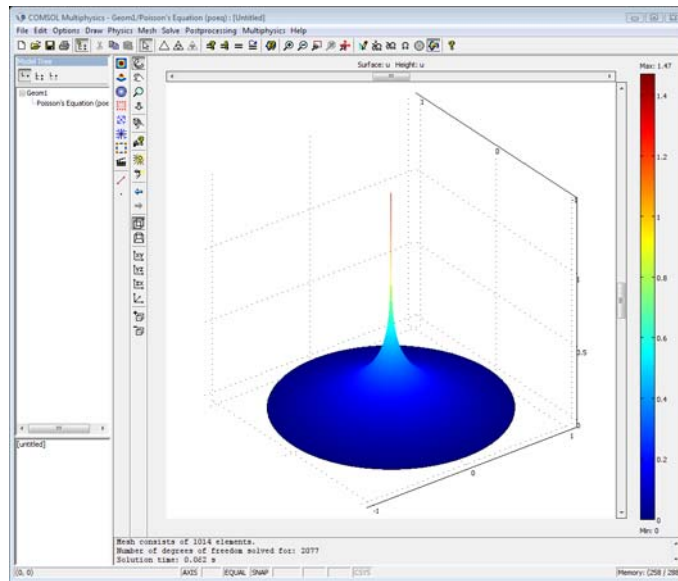
### **MESH GENERATION**

Because the solution has a singularity at the origin, use a much higher mesh resolution around that point:

- 1 From the **Mesh** menu choose **Free Mesh Parameters**.
- 2 Click the **Point** tab.
- 3 Select Point 3.
- 4 In the **Maximum element size** edit field type 0.001.
- 5 Click the **Remesh** button.
- 6 Click **OK**.

## COMPUTING THE SOLUTION

Click the **Solve** button on the Main toolbar.



## POSTPROCESSING AND VISUALIZATION

To view the solution as a 3D surface, click the **3D Surface Plot** button on the Plot toolbar.

The exact solution is unbounded at the origin, which complicates evaluation of the error. Because the solution is axisymmetric, you can compare the COMSOL Multiphysics solution to the exact solution on a ray from just outside the origin to the boundary.

- 1 From the **Postprocessing** menu choose **Cross-Section Plot Parameters**.
- 2 In the **Cross-Section Plot Parameters** dialog box, click the **Line/Extrusion** tab.
- 3 In the **Expression** edit field under **y-axis data** type  $u + \log(x^2) / (4 \cdot \pi)$ .
- 4 In the **Cross-section line data** area, type 0.02 in the **x0** edit field and type 1 in the **x1** edit field. Leave both **y0** and **y1** at the default value of 0.
- 5 Click **OK**.

Another way to check the solution's accuracy is to integrate the difference between the exact solution and the FEM solution over the domain. The result shows that although the local error close to the origin is quite large, the global error in the  $L^2$  norm is small.

# Sensitivity Analysis

This chapter describes the Sensitivity Analysis application mode and how you can use it in models to, for example, estimate modeling errors caused by uncertainties in material properties, or to predict the effect of a geometrical change.

# The Sensitivity Analysis Application Mode

The Sensitivity Analysis application mode is different from most other application modes in that it does not contain any physics of its own. Instead, it is a tool that makes it possible to evaluate the sensitivity of a model with respect to almost any variable.

## *Introduction*

---

Because the Sensitivity Analysis application mode does not contain any physics, it is not intended for use on its own. When you add Sensitivity Analysis to a multiphysics model, no new equations are introduced, and the set of solution variables remains the same. Instead, this application mode lets you introduce new *sensitivity variables*, which are stored in the solution vector but never solved for; the sensitivity variables are input quantities that you can use to control various aspects of your model's physics.

The purpose of the Sensitivity Analysis application mode is to compute sensitivities—which is the same thing as derivatives, or collectively the *gradient*—of various model quantities with respect to the sensitivity variables. To this end, the application mode lets you perform three distinct tasks:

- Select sensitivity variables and set their values
- Define an (optional) scalar objective function
- Compute the sensitivities efficiently using the sensitivity solver

## *Sensitivity Variable Selection*

---

In the Sensitivity Analysis application mode you can define an arbitrary number of sensitivity variables. These can either be fields represented on the finite element mesh, just like the solution variables, or global scalar quantities. Variables of the former type can represent a density, a distributed force, or the deformation of a boundary, to mention just a few possibilities. You can set up scalar sensitivity variables to control, for example, global material properties, rigid body displacements, or a wave number.

Note the duality of the sensitivity variables: on one hand, they are independent variables whose values are not affected by the solution process; on the other hand, they are degrees of freedom (DOFs) stored in the solution vector. When defining a



sensitivity variable, you must therefore supply its *initial value*, which the software uses to initialize the sensitivity variable DOFs; these remain fixed during the solution step.

### **SENSITIVITY VARIABLES AND FINITE ELEMENT REPRESENTATION**

The sensitivity analysis is always performed on the discretized system of equations. When using a sensitivity variable field represented on the finite element mesh, the sensitivities are therefore associated with individual sensitivity variable degrees of freedom rather than with the field value at each point. This makes it difficult to interpret the result.

For example, if you set up a subdomain sensitivity variable using a 1st-order Lagrange shape function representation to control the material density in your model, your solution will contain the sensitivity of the objective function with respect to the discrete density value at each node point in the mesh. Because each node influences the density in a small surrounding region, the size of which varies from node to node, the individual sensitivities are not directly comparable to each other.

When you display such subdomain sensitivity variables, the varying element size gives you a plot which is not smooth and must not be used to draw any conclusions about the physics and the effect of changing the physical field represented by the sensitivity variable. Some insight may, however, be gained by looking at the sensitivities divided by the mesh volume scale factor `dv01`. This makes the sensitivities in the plot comparable between different parts of the surface, but still not mathematically well defined. In particular, using discontinuous constant shape functions together with the division by `dv01` gives you a plot which is proportional to the true pointwise sensitivity.

If you intend to use the sensitivities in an automatic optimization procedure, as is done in the Optimization application mode available with the optional COMSOL Optimization Lab, the discrete nature of the sensitivities causes no additional complication. The optimization solver searches for optimum values of the discrete sensitivity variables using the discrete gradient provided by the sensitivity analysis.

### **GEOMETRICAL SENSITIVITY**

You can use the sensitivity variables directly to parameterize any aspect of the physics which is controlled by an expression. This applies for example to material properties, boundary conditions, loads and sources. However, the shape, size, and position of parts of the geometry cannot be changed as easily at solution time, and therefore require special attention.

Sensitivity variables cannot be used directly in the geometry description. Instead, you must set up your model using *ALE* and tie all physics to an *ALE frame* controlled by

a Parameterized Geometry application mode (in 2D only) or a Moving Mesh (ALE) application mode. You can then use sensitivity variables to control the mesh movement, effectively parameterizing the geometry.

See the chapter “Deformed Meshes” on page 453 for details on the Parameterized Geometry and Moving Mesh (ALE) application modes as well as on ALE in general. For an example, see “Example—Predicting the Effect of a Geometrical Change” on page 303 later in this chapter.

### *Scalar Objective Functions*

---

In many practical situations, you are interested in the behavior of a single scalar quantity when you change your control variables but not in the effect on the solution at every single point. A common example is when your final goal is to perform optimization with respect to some *objective function* (also known under different names such as *goal function*, *cost function*, or *quantity of interest*). Computing the sensitivity of a scalar objective function is in most cases cheaper than computing the sensitivity of every solution component.

The Sensitivity Analysis application modes lets you specify a scalar objective function, which is a sum of contributions integrated over subdomains, boundaries, and edges, and summed over points. You can also add any valid global expression to the composite objective function. Examples of scalar objective functions from different applied fields include quantities such as total power, efficiency, cup mixing temperature, backscatter, mass flow, and capacitance.

#### *The Principle of Virtual Work*

*Potential energy* has a special status among scalar objective functions, because its derivatives with respect to scalar sensitivity variables can in many cases be interpreted as (true or generalized) forces.

The optional AC/DC Module contains a Virtual Work predefined multiphysics coupling (for more information, see “Calculating Forces with Virtual Work” on page 55 of the *AC/DC Module User’s Guide*), which uses this principle to calculate electromagnetic forces as derivatives of the total electromagnetic energy stored in the system. The model “Electromagnetic Forces on Parallel Current-Carrying Wires” on page 8 of the *AC/DC Module Model Library* demonstrates the excellent accuracy of the method on a benchmark problem.

### Objective Functions for Frequency-Domain Modeling

Sensitivity analysis can be applied only when the objective function is a real-valued differentiable function of the sensitivity variables. This is usually not a very severe constraint, even for frequency-domain models where the solution variables are complex valued. The reason is that physical quantities of interest to the analyst are always real-valued, and if complex valued sensitivity variables are required, the real and imaginary parts can be treated separately.

Many common quantities of interest are, however, time averages which can be written on the form  $Q = \text{real}(a \cdot \text{conj}(b))$ , where  $a$  and  $b$  are complex-valued linear functions of the solution variables and therefore implicit functions of the sensitivity variables. The problem with this expression is that while  $Q$  is indeed a real-valued differentiable function of the sensitivity variables, it is not an analytical function of  $a$  and  $b$ . This complicates matters slightly because the sensitivity solver relies on partial differentiation and the chain rule.

While the partial derivatives of  $Q$  with respect to  $a$  and  $b$  are, strictly speaking, undefined, it can be proven that if they are chosen such that

$$Q(a + \delta a, b + \delta b) \approx Q(a, b) + \text{real}\left(\frac{\partial Q}{\partial a} \delta a + \frac{\partial Q}{\partial b} \delta b\right) \quad (10-1)$$

for any small complex increments  $\delta a$  and  $\delta b$ , the final sensitivities are evaluated correctly. The special function `real_dot(a, b)` is identical to `real(a*conj(b))` when evaluated but implements partial derivatives according to Equation 10-1. For that reason, use it in the definition of any time-average quantity set as objective function in a sensitivity analysis. For an example, see “Optimizing the Shape of a Horn” on page 34 of the *Acoustics Module Model Library* (note that this model requires both the Optimization Lab and the Acoustics Module).

### THE SENSITIVITY SOLVER

The sensitivity solver, which is intimately connected to the Sensitivity Analysis application mode, provides derivatives of model quantities with respect to the sensitivity variables. To evaluate these derivatives, or sensitivities, an auxiliary linear problem is solved, in addition to the original *forward model*. You have two options:

- Select the *forward sensitivity* method to evaluate the derivatives of all solution variables and an optional objective function.
- Select *adjoint sensitivity* to look only at derivatives of a scalar objective function.

Both methods have their benefits as will be explained below.

Sensitivity Analysis can be used together with all stationary and parametric standard solvers. The available solver settings are described in the section “Sensitivity Analysis and Optimization” on page 423 in the *COMSOL Multiphysics User’s Guide*. For technical details about the solution procedure, see “The Sensitivity Solver Algorithm” on page 541 in the *COMSOL Multiphysics Reference Guide*.

#### *Forward Sensitivity*

Use the forward sensitivity method to solve for the derivatives of all dependent variables, plus an optional scalar objective function, with respect to a small number of sensitivity variables. The forward method requires one extra linear system solution for each sensitivity variable.

The linear system that must be solved is the same as the last linearization needed for solving the forward model. Thus, when using a direct solver (for example, PARDISO) the extra work amounts only to one back-substitution per sensitivity variable DOF. The iterative linear and segregated solvers can reuse preconditioners and other data but must otherwise perform a complete solution each time.

#### *Adjoint Sensitivity*

The adjoint method solves for the derivatives of a single scalar objective function with respect to any number of sensitivity variables, requiring only one single additional linear system solution. In addition to the objective function gradient, the discrete adjoint solution is computed. This quantity represents the sensitivity of the objective function with respect to an additional generalized force applied as a nodal force to the corresponding solution component.

The auxiliary linear system is in this case the transpose of the last linearization needed for solving the forward model. The UMFPACK linear solver can always solve the transposed problem at the cost of a back-substitution, while remaining direct solvers need to do a new factorization if the problem is not symmetric or Hermitian. The iterative solvers can reuse most preconditioning information as can the segregated solver, which, however, loops over the segregated steps in reversed order.

### **POSTPROCESSING WITH THE SENS, ADJ, AND FSENS OPERATORS**

The solution returned from the sensitivity solver contains sensitivity information in addition to the forward model solution. The Sensitivity Analysis application mode defines postprocessing variables for direct access to the gradient of the objective function as well as that of the forward solution (if the forward method is used) or the adjoint solution (if the adjoint method is used).

You can also postprocess the sensitivity data using a number of dedicated operators in your postprocessing expressions:

**fsens(sensvar)** When you have specified an objective function,  $Q$ , the expression `fsens(sensvar)` evaluates to the derivative of  $Q$  with respect to sensitivity variable `sensvar`.

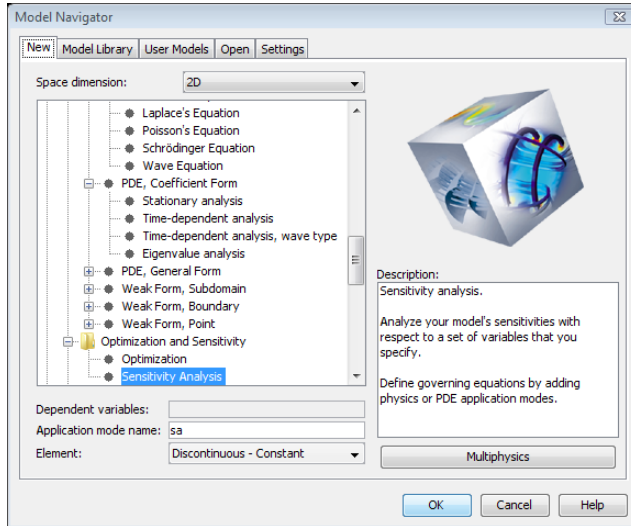
**sens(expr,sensvar)** After using the forward sensitivity method, `sens(expr,sensvar)` evaluates the expression `expr`, replacing the forward model solution  $u$  with its derivative with respect to `sensvar`. For example, `sens(u,p)` evaluates the derivative of forward model solution variable  $u$  with respect to sensitivity variable  $p$ . Note that although the operator allows general expressions, it only makes sense for expressions which are linear in  $u$ .

**adj(expr)** The adjoint sensitivity method computes and stores the adjoint solution with respect to the specified objective function. During postprocessing, the statement `adj(expr)` evaluates the expression `expr` with the adjoint solution substituted for the forward solution.

When working in a script environment, the corresponding sensitivity data are stored in the solution object fields `fem.sol.fsens`, `fem.sol.sens` and `fem.sol.adj`. The precise meaning of these fields is explained under `femsol` in the *COMSOL Multiphysics Reference Guide*.

## Model Navigator

You find the **Sensitivity Analysis** application mode in the **Optimization and Sensitivity** folder under **COMSOL Multiphysics**. No dependent variable names are required because the application mode does not add any new solution components.



*The Model Navigator with the Sensitivity Analysis application mode selected.*

## Application Mode Properties

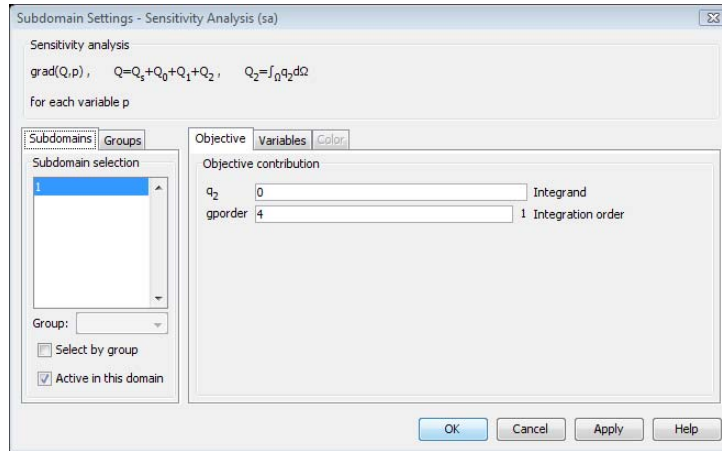
Unless your model contains more than one frame, there is only one property in the **Application Mode Properties** dialog box: using the **Default element type** list, select the default shape function for mesh-based sensitivity variables. If there are multiple frames, you can use the **Frame** list to select with respect to which coordinates spatial derivatives of sensitivity variables are defined.

## Subdomain, Boundary, and Edge Settings

The **Subdomain Settings**, **Boundary Settings**, and **Edge Settings** (in 3D only) dialog boxes all have the same structure. This section shows the subdomain settings only.

On the **Objective** page, you specify contributions to the objective function. The program integrates any expression you enter in the **Integrand** edit field over the domain and adds it to the global objective function. You can also change the **Integration order**

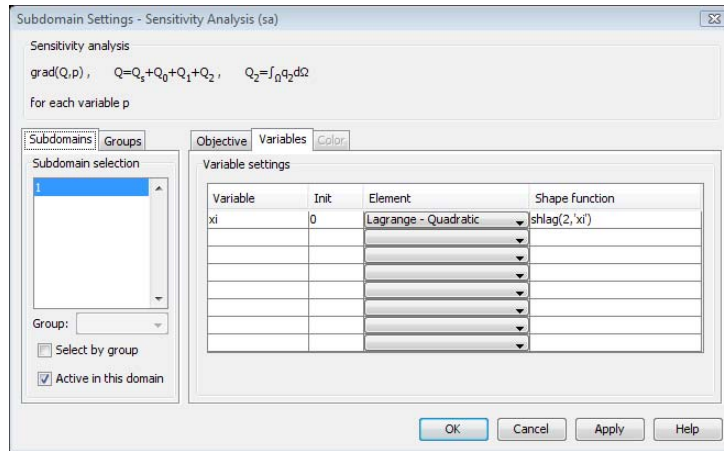
used for evaluating the integral. If there is more than one frame in the model, an additional **Frame** selection list appears. Select the frame in which you want integration to be performed.



*The Objective page of the Subdomain Settings dialog box.*

Use the table on the **Variables** page to define mesh-based sensitivity variables. Enter a variable name in the **Variable** column and its value in the **Init** column. The **Element** and **Shape function** columns are automatically filled in according to the preference setting in the **Application Mode Properties** dialog box. If you want a different discretization, select a predefined setting in the **Element** column, or modify the **Shape function** expression. The accepted syntax is the same as on the **Element** page in any other

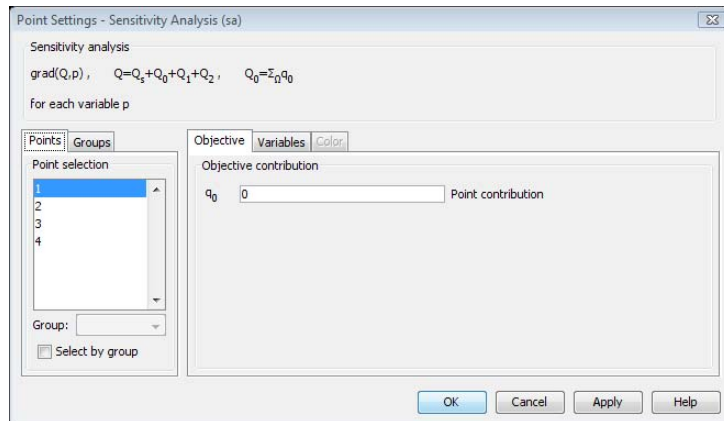
application mode (see “Finite Elements” on page 482 in the *COMSOL Multiphysics Reference Guide*).



*The Variables page of the Subdomain Settings dialog box.*

## Point Settings

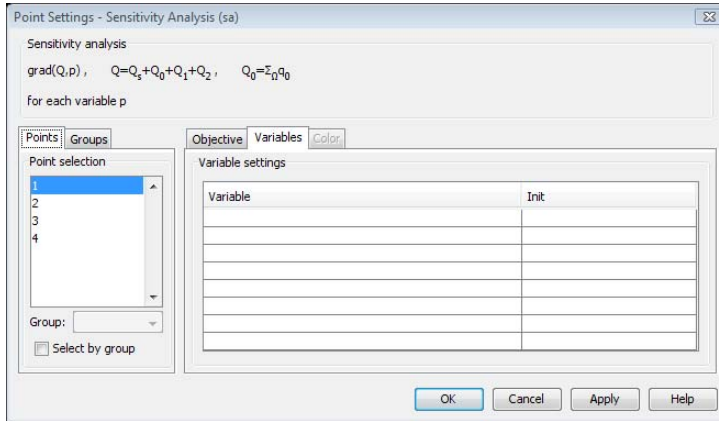
**Point Settings** differs from **Subdomain Settings** in that summation is used instead of integration. Therefore, no integration order or frame can be specified on the **Objective** page. The local **Point contribution** evaluated on each point adds directly to the global objective function.



*The Objective page of the Point Settings dialog box.*



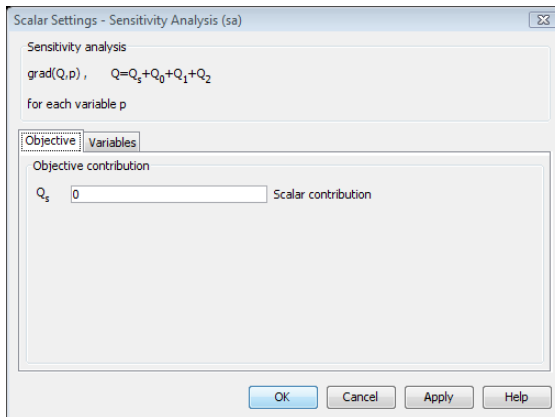
The **Variables** page is also simplified, compared to **Subdomain Settings**. You only need to enter a sensitivity variable name in the **Variable** column and its value in the **Init** column.



*The Variables page of the Point Settings dialog box.*

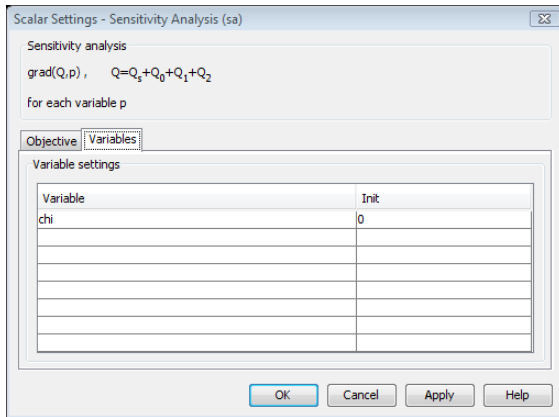
### Scalar Settings

On the **Objective** page of the **Scalar Settings** dialog box, you provide any contribution to the objective function for the current geometry that is independent of the position. Enter its value in the  $Q_s$  edit field.



*The Objective page of the Scalar Settings dialog box.*

On the **Variables** page, enter the names all of global scalar sensitivity variables in the **Variable** column and their values in the **Init** column.



*The Variables page of the Scalar Settings dialog box.*

# Example—Predicting the Effect of a Geometrical Change

In this section, the Sensitivity Analysis application mode is applied to an existing model to predict the effect of changing its geometry slightly.

## *Introduction*

---

The example “Stiffness Analysis of a Communication Mast’s Diagonal Mounting” on page 565 of the *COMSOL Multiphysics Model Library* shows how you can modify a 3D CAD model in order to improve its performance. In that case, the applied changes were based solely on the analyst’s experience with similar structures. A senior design engineer can sometimes reach acceptable performance after analyzing only a handful of designs, while an unexperienced analyst may have to spend a lot of time on failed attempts.

Usually, you can indeed improve a design by trial and error, but it is difficult to ensure that the price you pay—in this example, added weight and material costs—is as low as possible. With sensitivity analysis, you can find the most cost-efficient direction for a small modification and estimate the effect it will have before attempting an updated design.

## *Model Definition*

---

The original model simulates the deformation of a part of a communication mast (shown in Figure 10-1) under loads in the linear regime. A ratio of the part’s effective stiffness to the stiffness of an equal length of straight pipe is evaluated as a measure of its performance. Using sensitivity analysis together with a Moving Mesh application mode, you can predict what effect changing the dimensions of the end plate and the mount has on the part’s relative stiffness.

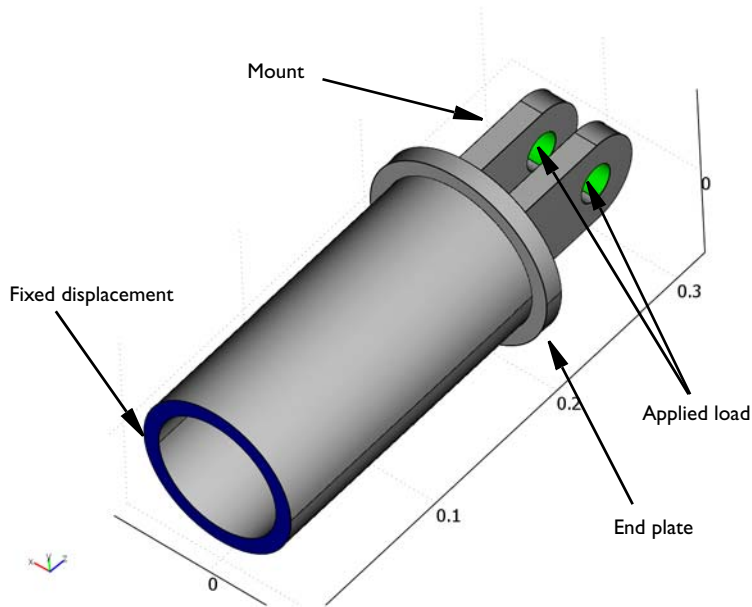


Figure 10-1: The original CAD design of the part.

From the designer's point of view, the material thickness of the end plate,  $t_1$ , and of the mount,  $t_2$ , are the most relevant parameters because they are easy to change both in the CAD system and on the production line. These quantities are, however, not directly comparable to each other because a unit change of  $t_1$  incurs a different cost—added weight and material use—compared to a unit change in  $t_2$ .

For a fair analysis, it is therefore more convenient to parameterize the model in terms of the masses  $m_1$  and  $m_2$  added to the plate and mount, respectively. The relation between added mass  $m_i$  and thickness change  $d_i$  for  $i = 1, 2$  is given by

$$m_i = \rho A_i d_i$$

where  $A_i$  is the area affected by the thickness change ( $m^2$ ) and  $\rho$  is the density of the material ( $kg/m^3$ ). You calculate the constant factor  $\rho A_i$  as an integration coupling variable  $MpL_i$  in the model.

As output from a sensitivity analysis using the relative stiffness  $S_R$  as objective function and the differential masses  $m_1$  and  $m_2$  as sensitivity variables, you get the partial derivatives

$$Q_i = \frac{\partial S_R}{\partial m_i}$$

For a modified geometry corresponding to small values of  $m_1$  and  $m_2$  you can therefore expect to see a change in the relative stiffness equal to  $\Delta S_R = Q_1 m_1 + Q_2 m_2$ . Note that this relation holds only for a small incremental change from the current configuration because the relative stiffness is clearly a nonlinear function of the thicknesses.

Now suppose that you want to select the best possible design update for a given added mass  $\Delta m = m_1 + m_2$  with the added condition that both  $m_1$  and  $m_2$  are nonnegative. It is not too difficult to realize that the best option is to take  $m_1 = \Delta m$ ,  $m_2 = 0$  if  $Q_1 > Q_2$ , and  $m_1 = 0$ ,  $m_2 = \Delta m$  otherwise.

The optimum stiffness for a given total mass of the structure can be sought by relaxing the nonnegativity condition for the updates and instead restricting the maximum change in  $m_1$  and  $m_2$  during one iteration. With the total mass as only constraint, you will find the optimum design at a point where  $Q_1 = Q_2$ . This follows strictly from the Karush-Kuhn-Tucker conditions but also from the simple fact that at such a point, the increased stiffness from adding mass to the plate is exactly cancelled by the decrease in stiffness from removing the same mass from the mount.

#### PARAMETERIZING THE GEOMETRY

In the Sensitivity Application mode, you declare *sensitivity variables*, which can be used to parameterize the physics of the model. The sensitivity variables can appear anywhere COMSOL Multiphysics accepts an expression containing the dependent variables. However, neither dependent variables nor sensitivity variables can be used directly to set dimensions in the geometry.

To evaluate the sensitivity of a model with respect to geometrical changes, the geometry must first be made an active part of the system of equations. You accomplish this by moving all physics onto an *ALE frame* controlled by a Moving Mesh (ALE) application mode (described in the section “The Moving Mesh Application Mode” on page 463). The Moving Mesh (ALE) application mode sets up an equation governing the position of the mesh nodes inside the domains, while the outer shape of the domain is controlled by boundary conditions.

When doing sensitivity analysis, these boundary conditions are quite simple: on fixed surfaces, set the mesh displacement to zero; on surfaces which may be modified, specify the displacement in terms of the sensitivity variables. In this particular case, where the

material thickness of the end plate and the mount can change, it is enough to set the normal displacement of these surfaces equal to the thickness change calculated from the corresponding added mass.

On surfaces adjacent to a domain with a parameterized normal displacement, it is preferable to restrict the mesh displacement to zero only in the normal direction to avoid an inconsistent constraint on the common edge. However, any remaining inconsistencies do not invalidate your results completely but only effectively modify the parametrization. In cases like this, when the purpose of the analysis is a rough estimate and guidance for a manual redesign, such minor errors in the sensitivities are unimportant.

Another potential source of errors must be checked more carefully, though. Changing the material thickness of the mount also changes the area where the loading is applied. In the Solid, Stress-Strain application mode, the load is specified as a given force per area. If you keep this number fixed when the thickness of the part changes, the total applied load also changes; when evaluating the relative stiffness of the composite part you must account for this effect.

Alternatively, you can keep the total force fixed and make sure that the applied force per area is calculated using a hole area that follows the parametrization of the geometry. It turns out that in the original model, the applied load is indeed calculated based on a hole area which is evaluated using integration coupling variables sensitive to deformation of the geometry.

### **CHOOSING FORWARD OR ADJOINT SENSITIVITY ANALYSIS**

By default, the Sensitivity Analysis application mode uses the *adjoint method*, which is more efficient than the *forward method* when the number of sensitivity variables is large. When there is only a handful of scalar parameters, as in this case, the forward method has the advantage that it returns the sensitivity of the entire solution with respect to the sensitivity variables in addition to the sensitivity of the objective function. This additional information can sometimes be important in itself, but more often it is useful for checking the model setup because it is easy to visualize.

### *Results and Discussion*

---

The analysis shows that when the thickness of the end plate is 12 mm and the material thickness in the mount is 15 mm, the sensitivities to adding mass to the two details are  $Q_1 = 0.27$  and  $Q_2 = 0.16$ , respectively. Because, apparently, adding mass to the end plate has more effect than adding it to the mount, the next redesign of the part should

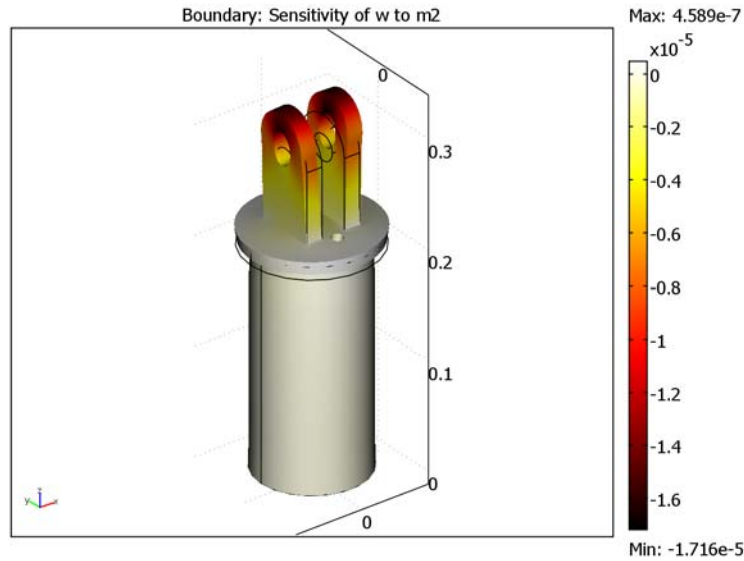
be fitted with a thicker plate. You might even consider decreasing the material thickness of the mount while adding to the plate to keep the weight of the part constant.

Note that these conclusions only hold for the current instance of the design. An experienced analyst quickly realizes that the compliance of the plate is due to bending action, while the mount is in almost pure tension. A plate's resistance to bending grows as its thickness cubed, while resistance to tension is proportional to the cross-sectional area. Therefore, as the thickness of the plate increases, its contribution to the compliance of the composite part decreases rapidly.

If you increase the thickness of the end plate too much, the compliance of the mount takes over as dominant factor in the overall behavior. Using sensitivity analysis, you can easily detect when this has happened, because it will lead to  $Q_2 > Q_1$ . As noted above, an optimum design for a given total mass is found when  $Q_1 = Q_2$ .

Because the sensitivity analysis was performed using the forward method, the derivatives of the solution with respect to the parameters  $m_1$  and  $m_2$  have also been stored. You can access this data during postprocessing by choosing them from the list of predefined quantities, or more generally using the syntax `sens(expr, var)`. For example, by plotting the expression `sens(w, m2)`, you can directly examine the local effect of a unit increase of the mount mass (see Figure 10-2). Note that, interestingly, increasing the material thickness in the mount reduces the bending of the end plate.

The reason is probably that, as the mount plates are getting thicker, they transfer the load closer to the edge of the end plate.



*Figure 10-2: According to the linear approximation, increasing the thickness of the end plate by 5 mm would reduce its bending and make the mount plates nearly parallel.*

It is also possible to use a composite expression such as  $u + 0.005 * m_{pL2} * \text{sens}(u, m1)$  to display a linear approximation of the actual deformation that would result from a 5 mm increase in the thickness of the end plate. In Figure 10-2 such expressions have been applied as boundary deformation data to let you can easily get an idea of the would-be effect of changing the geometry. Note that this is a very rough estimate because the bending stiffness as function of the plate thickness is far from linear.

---

**Model Library path:** COMSOL\_Multiphysics/Structural\_Mechanics/  
mast\_diagonal\_mounting\_sensitivity

---



## **MODEL NAVIGATOR**

You can use the **Model Navigator** to create a new model, open existing models and modify the application modes used in the current model. In this case, you have to do the two last. In the original model, the structural mechanics equations are defined in the reference coordinate system. For the sensitivity variables to have any effect on the structural simulation, you must transfer the latter to a moving frame defined by a Moving Mesh (ALE) application mode.

- 1 In the **Model Navigator**, go to the **Model Library** page and select **COMSOL Multiphysics>Structural Mechanics>mast diagonal mounting mphbin**.
- 2 Click **OK** to load the model.
- 3 From the **Multiphysics** menu, open the **Model Navigator** again.
- 4 In the **Application Modes** tree, select **COMSOL Multiphysics>Deformed Mesh>Moving Mesh (ALE)**, then click **Add**.
- 5 Select **Solid Stress-Strain (sld)** in the **Multiphysics** tree, then click the **Application Mode Properties** button.
- 6 In the **Application Mode Properties** dialog box, select **Frame (ale)** from the **Frame** list. Click **OK** to close the dialog box.
- 7 Select **COMSOL Multiphysics>Optimization and Sensitivity>Sensitivity Analysis** from the **Application modes** tree, then click **Add**.
- 8 Click **OK** to close the **Model Navigator**.

## **OPTIONS AND SETTINGS**

The original model already contains most of the variables needed for the sensitivity analysis. In particular, the objective function used to evaluate the performance of the design is defined as a global expression, `Stiffness_ratio`. To express the thickness change of the plate and mount in terms of added mass, you must add a pair of integration variables which evaluate the ratio of added mass to added thickness.

### *Integration Variables*

- 1 From the **Options** menu, select **Integration Coupling Variables>Boundary Variables**.
- 2 Select Boundary 4. Add a variable with the **Name** `MpL1` defined by the **Expression** `rho_sld`. Leave the other columns unchanged.
- 3 Select Boundaries 17 and 32, and create another variable with the **Name** `MpL2` defined by the same **Expression**, that is, `rho_sld`.

- 4 Click **OK** to close the dialog box.

#### *Global Expressions*

- 1 From the **Options** menu, choose **Expressions>Global Expressions**.
- 2 Add new variables **d1** and **d2** with defining expressions  $m1/MpL1$  and  $m2/MpL2$ , respectively.
- 3 Click **OK**.

### **PHYSICS SETTINGS**

The Moving Mesh (ALE) application mode is necessary to make the structural mechanics simulation sensitive to changes in the geometry. You calculate the sensitivity around the reference configuration, so all boundary displacements are set to zero, either directly or indirectly via the sensitivity variables.

#### *Application Mode Properties—Moving Mesh (ALE)*

- 1 From the **Multiphysics** menu, select the **Moving Mesh (ALE)** application mode.
- 2 Select **Physics>Properties** and set **Weak constraints** to **Off**.
- 3 Click **OK** to accept the change.

#### *Subdomain Settings—Moving Mesh (ALE)*

- 1 Choose **Physics>Subdomain Settings** to open the **Subdomain Settings** dialog box.
- 2 Select Subdomain 2, click the **No displacement** button, and then click **OK**.

#### *Boundary Settings—Moving Mesh (ALE)*

- 1 Open the **Boundary Settings** dialog box, then press Ctrl+A to select all boundaries.
- 2 From the **Coordinate system** list, select **Tangent and normal coord. sys. in reference mesh**.
- 3 Select the **dn** check box to lock the displacement in the normal direction. On most boundaries, the default zero displacement is the correct setting.
- 4 Select Boundary 3 and click the **dt1** and **dt2** check boxes to lock this boundary in all directions. This is necessary to prevent rigid-body rotation of the mesh.
- 5 Select Boundaries 4, 5, and 10 and set the **dn** field to **d1**.
- 6 Select Boundaries 19 and 27 and set **dn** to **-d1**.
- 7 Select Boundaries 17 and 32 and change the **dn** field to **d2**.
- 8 Click the **Pairs** tab and select Pairs 2 and 3.
- 9 Deselect the **Active pair** check box to decouple the pairs.
- 10 Click **OK**.

### *Scalar Settings—Sensitivity Analysis*

The sensitivity variables are global scalar parameters. You set them up and specify their values in the **Scalar Settings** dialog box, where you can also enter scalar contributions to the objective function.

- 1 Activate the **Sensitivity Analysis** application mode by selecting it from the **Multiphysics** menu or in the **Model Tree**.
- 2 Choose **Physics>Scalar Settings** to open the **Scalar Settings** dialog box.
- 3 In the **Scalar contribution** edit field on the **Objective** page, enter `Stiffness_ratio`.
- 4 Click the **Variables** tab. In the **Name** column, enter the variable names `m1` and `m2` on separate lines in the table, both with **Init** value 0.
- 5 Click **OK** to close the dialog box.

### **COMPUTING THE SOLUTION**

Apart from a few special properties—the choice between forward and adjoint method being the most important—the sensitivity solver accepts the same solver settings as the standard stationary solver. In this case, the mesh displacements must be a part of the solution, but because they are independent of the physics, it is possible to solve them in a separate step before solving the structural displacements. To do this, use the segregated solver. You can also solve the coupled problem efficiently using an iterative solver as described below.

Correct scaling is important for accurate sensitivity evaluation. The structural displacements are on the order of  $10^{-5}$  m, while the mesh coordinates are on the order of  $10^{-1}$  m. You can specify these scale factors manually to improve the stability of the solution.

- 1 Choose **Solve>Solver Parameters** or click the corresponding button on the Main toolbar to open the **Solver Parameters** dialog box.
- 2 From the **Linear system solver** list, select **GMRES**, and leave the default **Geometric multigrid** preconditioner.
- 3 Turn on the Sensitivity solver by selecting the **Optimization/Sensitivity** check box, then click the **Optimization/Sensitivity** tab.
- 4 Change the **Sensitivity method** from **Adjoint** to **Forward**.
- 5 On the **Advanced** page, in the **Scaling of variables** frame, select **Manual** scaling and enter `x 1e-1 y 1e-1 z 1e-1 u 1e-5 v 1e-5 w 1e-5` in the **Manual scaling** edit field.
- 6 Click **OK** to close dialog box.

- 7 From the **Solve** menu, select **Update Model** to initialize the Moving Mesh (ALE) application mode.
- 8 Click the **Solve** button on the Main toolbar.

## POSTPROCESSING AND VISUALIZATION

To display the sensitivity of the deformation field and produce Figure 10-2, do the following:

- 1 Click **Plot Parameters** on the Main toolbar to open the corresponding dialog box.
- 2 On the **Boundary** page, select **Sensitivity Analysis (sa)>Sensitivity of w to m2** from the **Predefined quantities**.
- 3 Change **Color map** to **hot**.
- 4 Go to the **Deform** page and click the **Boundary Data** tab in the **Deformation data** frame.
- 5 Set the edit fields according to the following table:

FIELD	EXPRESSION
X component	$u+0.005*MpL1*sens(u,m1)$
Y component	$v+0.005*MpL1*sens(v,m1)$
Z component	$w+0.005*MpL1*sens(w,m1)$

- 6 Click **OK** to close the dialog box and display the new plot.  
Both the objective function and the sensitivity variables are global in this case, so you want to evaluate rather than plot them.
- 7 From the **Postprocessing** menu, select **Data Display>Global**.
- 8 As a first check, select **Sensitivity Analysis (sa)>Objective** from the **Predefined quantities** list, then click **Apply** to verify that the stiffness ratio is still 0.52 as in the original model.
- 9 In the same manner, evaluate the predefined quantities **Sensitivity Analysis (sa)>Objective sensitivity to m1** and **Sensitivity Analysis (sa)>Objective sensitivity to m2**. The values should be about 0.27 and 0.16, respectively.

The clear conclusion you can draw from these results is that adding thickness to the plate has better effect on the overall stiffness compared to adding the same mass to the mount.



# Optimization

If you have a license for the Optimization Lab, you get access to the Optimization application mode in the COMSOL Multiphysics graphical user interface. We have designed this application mode to facilitate setting up and solving constrained optimization and inverse-modeling problems based on multiphysics models.

This chapter describes the Optimization application mode's user interface and illustrates some modeling guidelines through a tutorial example model. You find additional optimization models for a variety of application areas in the *COMSOL Multiphysics Model Library* as well as in the Model Libraries for the add-on modules; look for the symbol “□” in the Model Library Guide tables.

Further background on optimization is available in the *Optimization Lab User's Guide*.

---

**Note:** The Optimization application mode requires the Optimization Lab.

---

# The Optimization Application Mode

## *Introduction*

---

Numerical optimization can be described as the art of finding an optimum set of parameters, or *optimization-variable* values, that minimize an *objective function* subject to a number of constraints. The objective function (also known under different names such as *goal function*, *cost function*, or *quantity of interest*) can be an explicit function of the optimization variables, or—which is a more interesting case—an output quantity from a simulation parameterized by the optimization variables.

While the Optimization application mode can indeed solve classical pure optimization problems, its real strength is its ability to compute accurate gradients also when the objective function and constraints depend on the solution to a multiphysics model. This enables the use of an efficient gradient-based optimization algorithm, reducing the number of multiphysics model evaluations required to find an optimal solution.

The Optimization application mode is in many ways similar to the Sensitivity Analysis application mode (see “The Sensitivity Analysis Application Mode” on page 292). Most importantly, they define variables and compute gradients in the same way. Moreover, neither application mode defines any physics of its own. Instead, you usually build an optimization model on top of an existing multiphysics model.

To define and solve a complete optimization problem, you need to perform four tasks in addition to those that apply when solving a standard PDE problem:

- Define a scalar objective function
- Select optimization variables
- Set additional constraints
- Compute optimal values of the optimization variables

The Optimization application mode provides an interface for handling all four tasks.

## *Optimization Problem Formulation*

---

The Optimization application mode is built around a general minimization problem formulation, where you supply an objective function to be minimized and constraints on the optimization variables. On top of this, the complete multiphysics model and its

constraints serve as an additional condition on the optimization problem. For this reason, the procedure is sometimes referred to as *PDE-constrained optimization*.

After you have set up a complete optimization model, the goal of the optimization process can be stated roughly as “find the values of the optimization variables that minimize the objective function under the condition that  $u$  is a solution to the multiphysics model and no constraints are violated.” More formally, this can be stated as follows:

$$\begin{aligned} \min_{\xi} \quad & Q(u, \xi) \\ & L(u, \xi) = 0 \\ & \text{lb}_P \leq P(\xi, u) \leq \text{ub}_P \\ & \text{lb}_\Psi \leq \Psi(\xi) \leq \text{ub}_\Psi \end{aligned} \quad (11-1)$$

The objective function  $Q(u, \xi)$  is a function of the optimization variables  $\xi \equiv \{\xi_i\}$  both directly and indirectly through  $u$ , which is the solution to the general multiphysics problem  $L(u, \xi) = 0$ . The objective function is defined as the sum of integral contributions from all dimensions  $k$  in the  $d$ -dimensional multiphysics model, and a general scalar term:

$$Q(u, \xi) = Q_s(u, \xi) + \sum_{k=0}^d \int_{\Omega_k} Q_k(u, \xi) dV_k$$

The implicit dependence of  $u$  on  $\xi$  means that the entire multiphysics problem can be seen as a constraint on the minimization problem. Additional constraints can be of two types:

- Integral constraints
- Pointwise constraints

Integral constraints assign lower and upper bounds to quantities  $P_j$  with general integral contributions of the same type as the objective function,

$$\text{lb}_{P,j} \leq P_j(\xi, u) \leq \text{ub}_{P,j}, \quad P_j \equiv P_{s,j}(u, \xi) + \sum_{k=0}^d \int_{\Omega_k} P_{k,j}(u, \xi) dV_k$$

for  $j = 1, \dots, n_{ic}$ , where  $n_{ic}$  is the number of integral constraints.

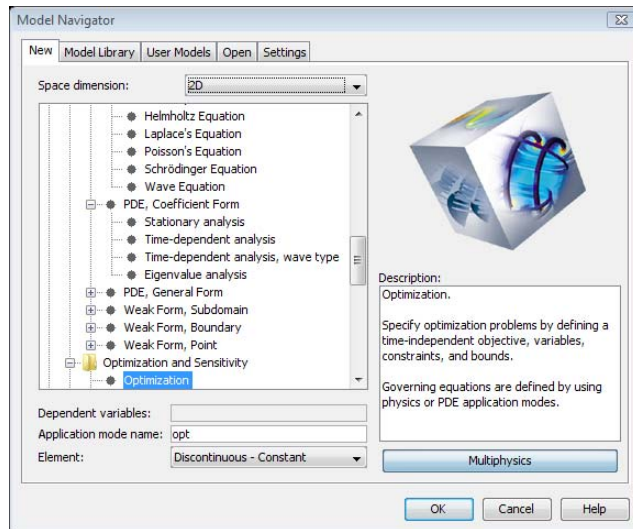
Pointwise constraints, on the other hand, apply in any single dimension (0, ...,  $d$  or scalar) separately. They specify lower and upper bounds for explicit functions of the optimization variables:

$$\text{lb}_{\Psi, i} \leq \Psi_i(\xi) \leq \text{ub}_{\Psi, i}, \quad i = 1, \dots, n_{\text{pc}}$$

where  $n_{\text{pc}}$  is the total number of pointwise constraints in all dimensions. They are less general than the integral constraints because they cannot depend on  $u$ , but they are more efficient to compute. It is good practice to use pointwise constraints to restrict the space of allowed optimization variable vectors  $\xi$  as far as possible before calling the optimization solver.

### Model Navigator

The Optimization application mode is available in all space dimensions. To include it in your model, open the **Model Navigator**, select the appropriate geometry from the **Space dimension** list, and then choose **COMSOL Multiphysics>Optimization and Sensitivity>Optimization** from the **Application Modes** tree.



*The Optimization application mode in the Model Navigator.*

Before clicking **OK** (or **Add** if you have previously clicked the **Multiphysics** button), you have the option of modifying the name in the **Application mode name** list (the default name is **opt**). In the **Element** type list, you can specify which shape functions to use.

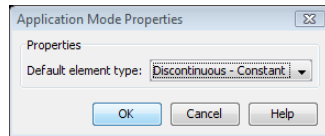


The setting in this list determines the default option when you define your design variables, so you can easily change it later if necessary.

### *Application Mode Properties*

---

In the **Application Mode Properties** dialog box—which you reach from the **Model Navigator** or by choosing **Properties** from the **Physics** menu—it is possible to specify the default element type in the list with the same name.



*The Application Mode Properties dialog box.*

If your model contains a Moving Mesh (ALE) or Parameterized Geometry application mode, you can also select the frame for your Optimization application mode.

### *Subdomain Settings*

---

In the **Subdomain Settings** dialog box for the Optimization application mode, you specify the objective function and any constraints that apply at the subdomain level. The settings are arranged in the following categories, to each of which corresponds a separate dialog-box page:

- Objective
- Integral Constraints
- Pointwise Constraints
- Variables

The following subsections describes these pages in turn. The presentation and the dialog-box snapshots apply directly to the case of a 3D geometry. However, the lower-dimensional cases are completely analogous.

#### **THE OBJECTIVE PAGE**

In 3D, the most general objective function takes the form

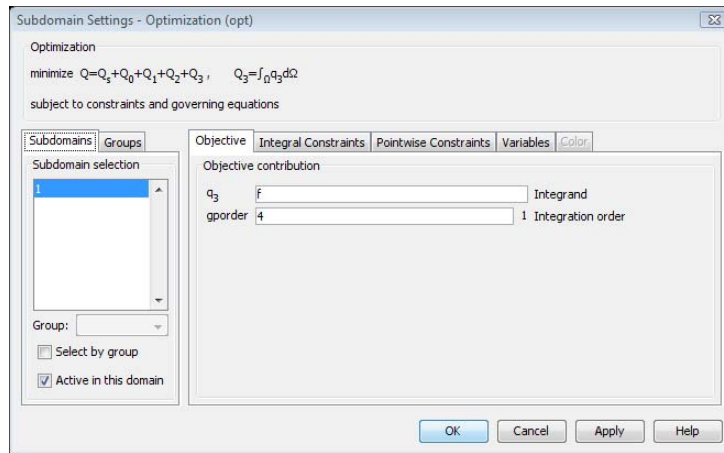
$$Q = Q_s + Q_0 + Q_1 + Q_2 + Q_3 \quad (11-2)$$

where  $Q_s$  denotes the scalar contribution, and  $Q_k$  denotes the contribution from domains of dimension  $k$  ( $k = 0, \dots, 3$ ). You specify these contributions on the **Objective** page of the **Scalar Settings**, **Point Settings**, **Edge Settings**, **Boundary Settings**, and **Subdomain Settings** dialog boxes, respectively.

Focusing first on the subdomain-level contribution, it is given by

$$Q_3 = \sum_l \int_{\Omega_k} (q_3)_l d\Omega_l \quad (11-3)$$

where the sum runs over all subdomains in the geometry. To define  $Q_3$ , provide the integrand  $(q_3)_l$  for every applicable subdomain  $l$  by selecting the subdomain from the **Subdomain selection** list and entering the corresponding integrand in the  $q_3$  edit field. (If a single integrand expression applies to several subdomains, you can of course select them concurrently.) In the **gporder** edit field you can, optionally, alter the integration order.



### THE INTEGRAL CONSTRAINTS PAGE

In 3D, the most general integral constraint takes the form

$$lb \leq P(\xi, u) \leq ub \quad P = P_s + P_0 + P_1 + P_2 + P_3 \quad (11-4)$$

where  $P_s$  denotes the scalar contribution and  $P_k$  denotes the contribution from domains of dimension  $k$  ( $k = 0, \dots, 3$ ). You specify the corresponding contributions on the **Integral Constraints** page of the **Scalar Settings**, **Point Settings**, **Edge Settings**, **Boundary Settings**, and **Subdomain Settings** dialog boxes, respectively.

---

**Note:** Define the lower and upper bounds, lb and ub, in the **Scalar Settings** dialog box.

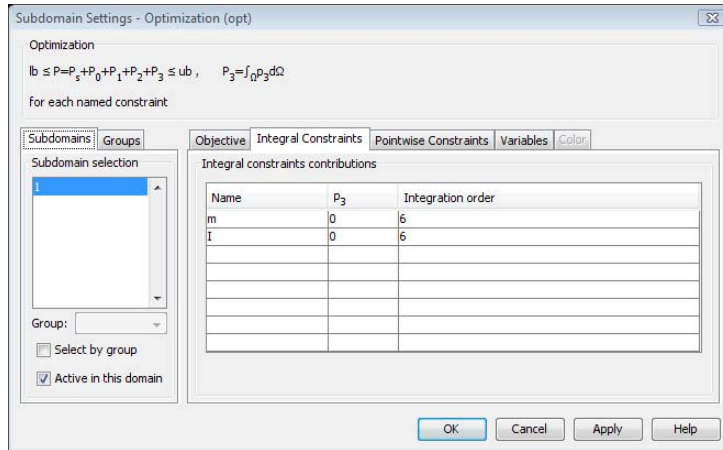
---

You can impose an arbitrary number of integral constraints,  $P_{(1)}, \dots, P_{(N)}$ , identifying each one by a unique name that you supply in the **Name** column on the **Integral Constraints** page. As soon as you have defined any one of the terms on the right side of the equation for  $P$  in Equation 11-4, the constraint automatically appears on the **Integral Constraints** page at all other levels.

To define a constraint contribution at the subdomain level in 3D,

$$P_3 = \sum_l \int_{\Omega_k} (p_3)_l d\Omega_l \quad (11-5)$$

specify the integrand  $(p_3)_l$  for every subdomain  $l$  on which the constraint applies by selecting the subdomain from the **Subdomain selection** list and entering the corresponding integrand in the  **$p_3$**  column. Additionally, you need to specify the **Integration order** in the third column of the **Integral constraints contributions** table.

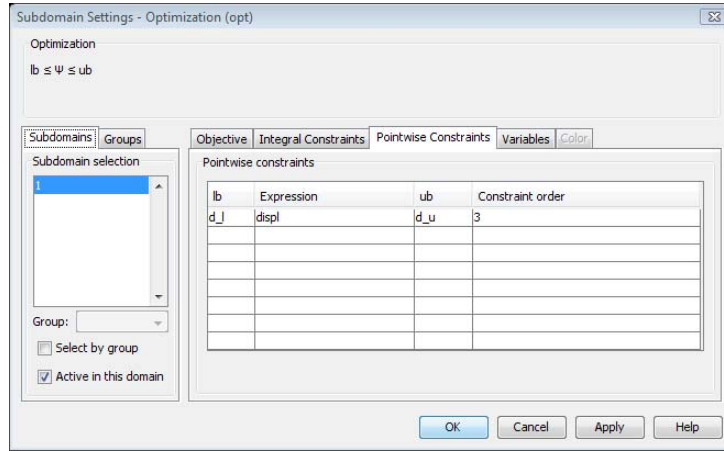


### THE POINTWISE CONSTRAINTS PAGE

On the **Pointwise Constraints** page of the **Subdomain Settings** dialog box, you specify constraints that apply at each point in a subdomain (or a set of subdomains)  $\Omega$ :

$$lb(x) \leq \Psi(\xi, x) \leq ub(x), \quad x \in \Omega$$

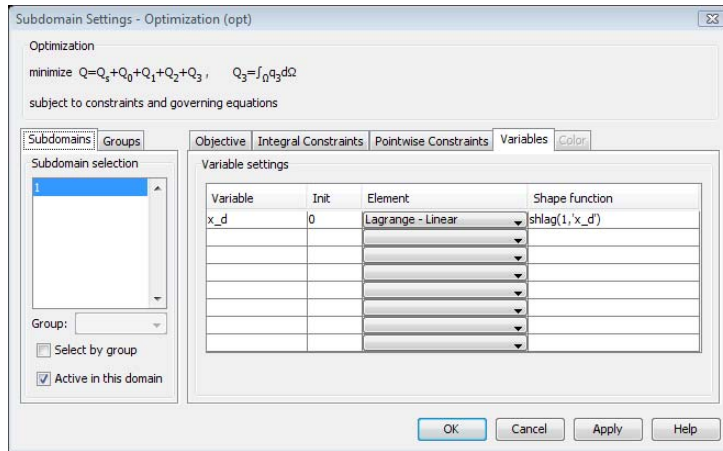
For each constraint, supply the lower and upper bounds,  $lb(x)$  and  $ub(x)$ , in the **lb** and **ub** columns, respectively, and the constraint expression  $\Psi(\xi, x)$  in the **Expression** edit field. In addition, you can specify the order of the constraint in the **Constraint order** edit field. The constraint is enforced discretely at Lagrange points of the specified order.



### THE VARIABLES PAGE

On the **Variables** page, you define the variables for your optimization problem. For each optimization variable, type a name in the **Variable** column and an initial value in the **Init** column. To specify the shape function, either select one of the options in the **Element** drop-down list or type it in the **Shape function** column using the syntax

described in the *COMSOL Multiphysics Reference Guide* (see `shdisc` on page 437 and `shlag` on page 441 for discontinuous and Lagrange elements, respectively).



## Boundary Settings

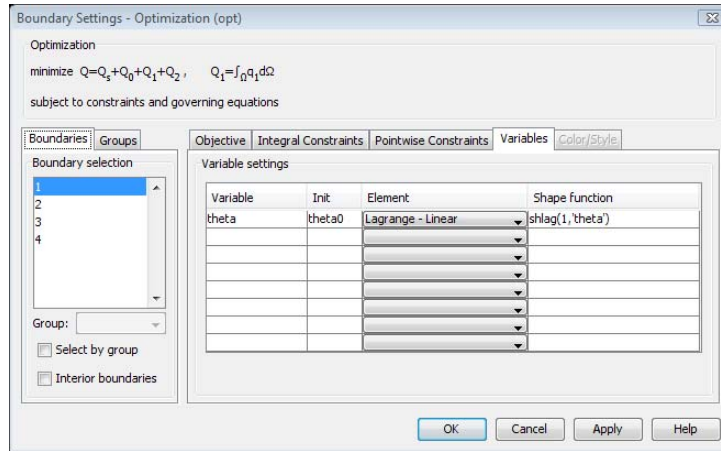
The **Boundary Settings** dialog box for the Optimization application mode contains the same pages as the **Subdomain Settings** dialog box.

On the **Objective** page you define the term  $Q_2$  in Equation 11-2 in a way completely analogous that for defining  $Q_3$  discussed in the previous section. In other words, you supply the integrands,  $(q_2)_l$ , and (optionally) the integration order. See “The Objective Page” on page 317 for further details.

Similarly, on the **Integral Constraints** page, you define the term  $P_2$  in Equation 11-4 by supplying the same data as at the subdomain level; see “The Integral Constraints Page” on page 318.

The **Pointwise Constraints** page, too, has the same function and design as the corresponding page on the **Subdomain Settings** dialog box described on page 319.

Finally, the **Variables** page is also completely analogous to its counterpart on the **Subdomain Settings** dialog box. Therefore, consult “The Variables Page” on page 320 for additional information.



### *Edge Settings (3D Only)*

On the Optimization application mode’s **Edge Settings** dialog box you specify the terms  $Q_1$  and  $P_1$  in Equation 11-2 and Equation 11-4, respectively. Its design and usage are completely analogous to those of the **Subdomain Settings** dialog box. Therefore, see the section “Subdomain Settings” on page 317 for details.

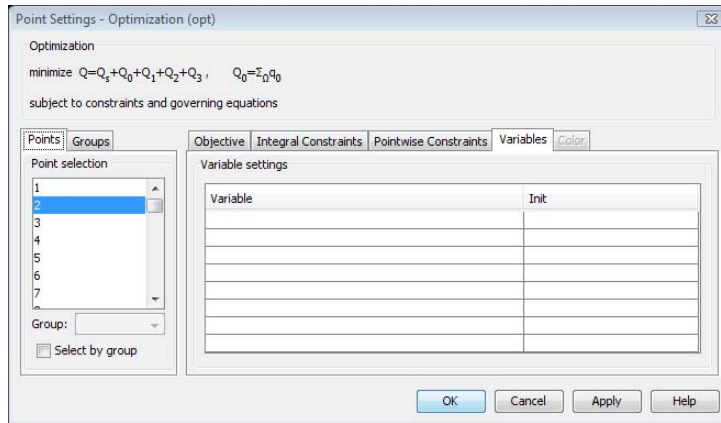
### *Point Settings*

Go to the **Objective** page of the **Point Settings** dialog box to specify any nonzero point contributions to the objective function,  $Q$  (the term  $Q_0$  in Equation 11-2). Note that these contributions are summed rather than integrated over the points.

On the **Integral Constraints** page, you specify the term  $P_0$  for constraints of the general form given in Equation 11-4. Again, the contribution is a sum rather than an integral.

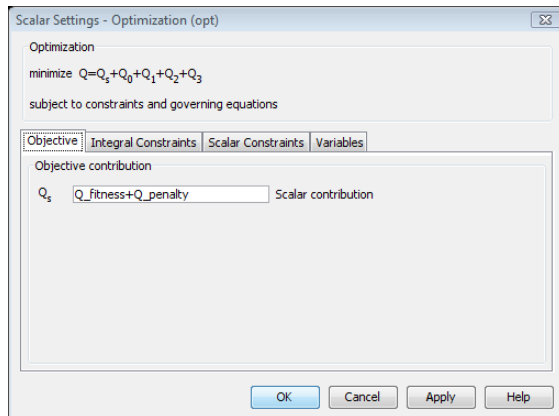
Constraints that are independent of the PDE solution,  $u$ , and must hold at each point separately, can be defined on the **Pointwise Constraints** page.

On the **Variables** page, you can specify optimization variables defined at the point level and their initial values.

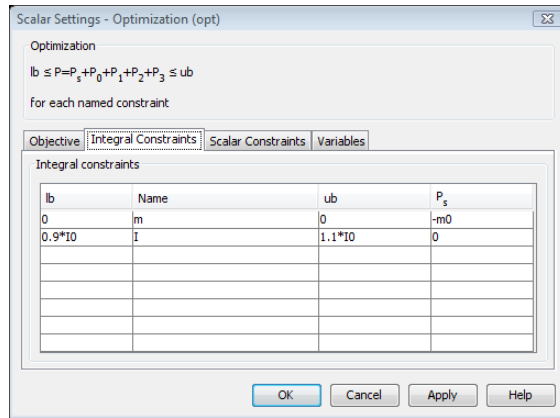


### *Scalar Settings*

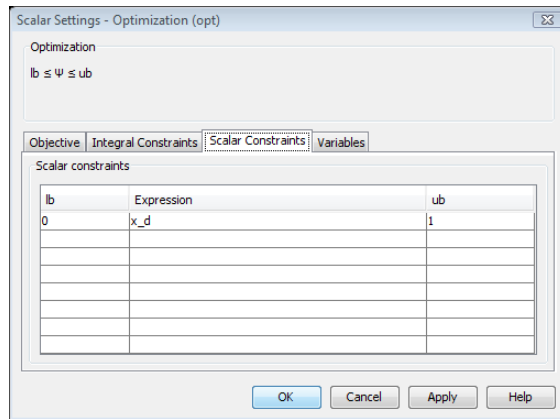
On the **Objective** page of the **Scalar Settings** dialog box, you provide any contribution to the objective function that is independent of the geometry and position. In Equation 11-2 this corresponds to the term  $Q_s$ . Enter its value in the  $Q_s$  edit field.



On the **Integral Constraints** page, you specify scalar contributions as well as the lower and upper bounds for your integral constraints.



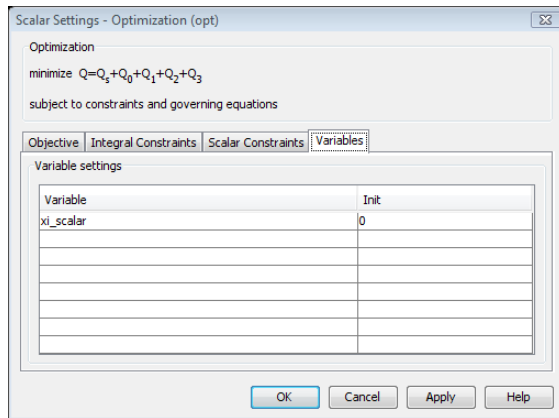
On the **Scalar Constraints** page you can define any number of scalar constraints by their expressions and lower and upper bounds.



Note that the constraints you specify on this page are of the type “pointwise constraints” according to the classification given in the section “Optimization Problem Formulation” on page 314. Thus, they cannot depend on the solution to the PDE problem, but they are more efficient to compute than integral constraints with only a scalar term.



Finally, on the **Variables** page, you specify any design variables that are independent of the position and geometry. Define the variable names in the **Variable** column and the corresponding initial values in the **Init** column.



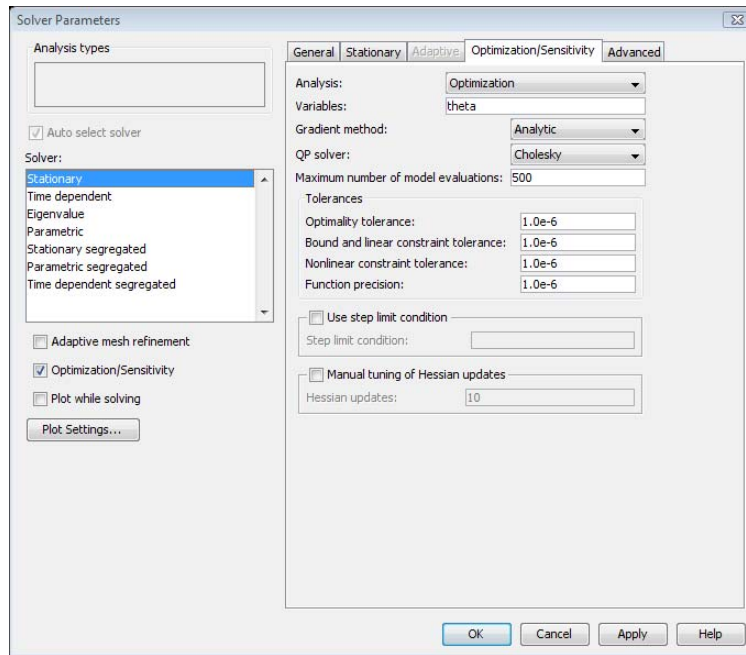
## *Solver Settings*

### **ACTIVATING THE OPTIMIZATION PROBLEM SOLVERS**

To solve the optimization problem you have specified, you need to do some settings in the **Solver Parameters** dialog box before starting the solver.

- 1 Open the **Solver Parameters** dialog box from the **Solve** menu or by clicking the corresponding button on the Main toolbar.
- 2 Select the **Optimization/Sensitivity** check box.
- 3 Optionally, select the **Plot while solving** check box. After each model evaluation, COMSOL Multiphysics then displays a plot in a separate figure window using the current settings in the **Plot Parameters** dialog box. In addition, the value of the objective function appears in place of the title above the plot. Select this option if you want to follow the optimization solver's progress with regards to some specific solution properties. Note that you can always monitor the objective function's value on the **Log** page of the **Progress** dialog box.
- 4 Click the **Optimization/Sensitivity** tab.
- 5 By default, the **Analysis** is set to **Sensitivity** so change this setting to **Optimization** to solve an optimization problem. It is, however, good practice to first solve once using the sensitivity solver, to check that the PDE model and optimization variables are set up correctly.

- 6 The names of any optimization variables you have defined appear automatically in the **Variables** edit field in the form of a space-separated list. If you only want to optimize with respect to a subset of these variables you can edit this entry.



This completes the optimization-problem solver settings that you always have to do. In some cases, you may need to perform some additional tuning as described in the subsequent subsections.

## OPTIMIZATION SOLVER SETTINGS

Often you do not need to worry about the settings described in this subsection, but they can sometimes be useful or necessary to adjust. For background information and further details, see the sections “Formulating Optimization Problems” on page 8 and “Gradient-Based Solver Properties” on page 64 of the *Optimization Lab User’s Guide*.

### Gradient Evaluation Method

In the **Gradient method** list, you can choose between **Analytic** (default) and **Numeric**. With the **Analytic** option set, the gradient is computed by the sensitivity solver using the adjoint method (see “The Sensitivity Solver” on page 295). For the **Numeric** option, the gradient is approximated by finite differences and computed by repeated

evaluation of the objective function. The numeric method can be more robust in some cases, but must only be used when the number of degrees of freedom for the optimization variables is small. In general, the default setting is more accurate and efficient.

#### *QP Solver Algorithm*

In the **QP solver** list you can specify which strategy the optimization solver should use when solving quadratic-programming subproblems. The following options are available:

- **Cholesky** (default)—For a small number of optimization-variable degrees of freedom (less than 1000) it is most efficient to compute and store a complete Cholesky factorization of the reduced Hessian matrix. If the number of degrees of freedom is large (higher than 1000) the QP solver will automatically switch to the **Quasi-Newton** method.
- **Conjugate gradients**—This is an appropriate choice for problems with many optimization-variable degrees of freedom. Because an iterative linear solver is used to generate step directions in the QP solver, no factorization of the reduced Hessian needs to be stored.
- **Quasi-Newton**—If the number of degrees of freedom is large, a partial Cholesky factor can be maintained instead of the complete factorization. This uses less memory than the **Cholesky** method, but should normally be more robust than **Conjugate gradients**.

For details, see “Qpsolver” on page 82 of the *Optimization Lab User’s Guide*.

#### *Maximum Number of Model Evaluations*

In the **Maximum number of model evaluations** edit field, you can set a limit on the number of times the solver evaluates the objective function before terminating, even if the optimality criterion has not been met. The default value is 500.

By setting this parameter to a smaller value and calling the optimization solver repeatedly, you can study the convergence rate and stop when further iterations with the optimization solver no longer have any significant impact on the value of the objective function.

#### *Tolerances*

In the **Tolerances** area, it is possible to adjust four related parameters:

- **Optimality tolerance**—A parameter used by the linear and quadratic solvers to determine, on the basis of the reduced-gradient size, whether optimality has been

reached (for details, see “Opttol” on page 77 of the *Optimization Lab User’s Guide*). For small values of the objective function, 0.01 times the parameter value is used as absolute tolerance. The default value is  $10^{-6}$ .

- **Bound and linear constraint tolerance**—This parameter controls when linear constraints are considered to be satisfied (for details, see “Feastol” on page 69 of the *Optimization Lab User’s Guide*). The default value is  $10^{-6}$ .
- **Nonlinear constraint tolerance**—This parameter specifies the accuracy with which the nonlinear constraints should be satisfied (for details, see “Majfeastol” on page 74 of the *Optimization Lab User’s Guide*). The default value is  $10^{-6}$ .
- **Function precision**—This parameter gives a measure of the relative accuracy with which the objective function and constraints can be evaluated (for details, see “Funcprec” on page 70 of the *Optimization Lab User’s Guide*). For function values smaller than 1, the parameter is used as an absolute precision. The default value is  $10^{-6}$ .

The optimality tolerance is the most important setting. In particular, it can play tricks on you if your objective function or your optimization variables are badly scaled. Preferably, the objective function and scalar constraints, as well as the optimization variables, should be of the same order, and ideally on the order of 1.

If the optimization solver reports a converged solution after just a few iterations, try to restart it with a tighter tolerance to make sure it has actually found the solution. If, on the contrary, it seems to iterate forever—despite the value of the objective function having converged (check the output on the **Log** page in the **Progress** window)—chances are that the tolerance value is too strict.

Note that the function precision specifies the expected stability of the numerical model rather than its accuracy as a model of physical reality. When using a direct linear solver on a linear model, the function precision should be set on the order of the inverse of the condition number. For a nonlinear or iterative solver, you can expect the precision to be of the same order as the solver tolerances.

#### *Use Step Limit Condition*

Selecting the **Use step limit condition** check box activates the **Step limit condition** edit field. In this edit field, you can enter an expression that tells the optimization solver to reduce the step length in the current line search because it has reached a point in the parameter space where the PDE problem is not well defined.

A typical example is when a mesh element becomes inverted during geometry optimization using a Moving Mesh ALE application mode. The step limit condition

that identifies this situation takes the form `minqual1_ale-0.05`, where 0.05 is a threshold value for the mesh quality. This step limit condition has a direct analog in the stop condition for the time-dependent and parametric solvers discussed in the section “Stop Condition” on page 468; see that section for additional details.

Note that the step limit condition should only be used as a last resort to keep the optimization solver in a feasible region. If possible, use instead pointwise constraints on the optimization variables to enforce the condition.

When the step limit condition is violated, the line-search step is reduced until an acceptable point is found. However, because no Jacobian is computed for the step limit condition, there is no mechanism to prevent the solver from immediately attempting another step in the same infeasible direction. As a result, it may get stuck at the same point without converging until the maximum number of model evaluations is reached or you stop the iteration manually using the Stop button in the **Progress** window.

#### *Manual Tuning of Hessian Updates*

Select the **Manual tuning of Hessian updates** check box to activate the **Hessian updates** edit field. This parameter (with default value 10) controls the quality of the approximate Hessian. Reducing its value can lead to shorter solution times since each QP subproblem becomes less expensive to solve. Increasing the number of Hessian updates, on the other hand, improves the robustness of the outer iterations. For details, see “Hessupd” on page 72 of the *Optimization Lab User’s Guide*.

#### **PLOT WHILE SOLVING**

To plot the multiphysics model solution in a separate window after each objective function evaluation, select the **Plot while solving** check box. The current settings in the **Plot Parameters** dialog box are then transferred to the solver. This lets you follow the progress of the optimization procedure via a suitably selected model quantity. Note that the current objective function value is displayed in the figure title.

# Example—Minimizing the Flow Velocity in a Microchannel

## *Introduction*

---

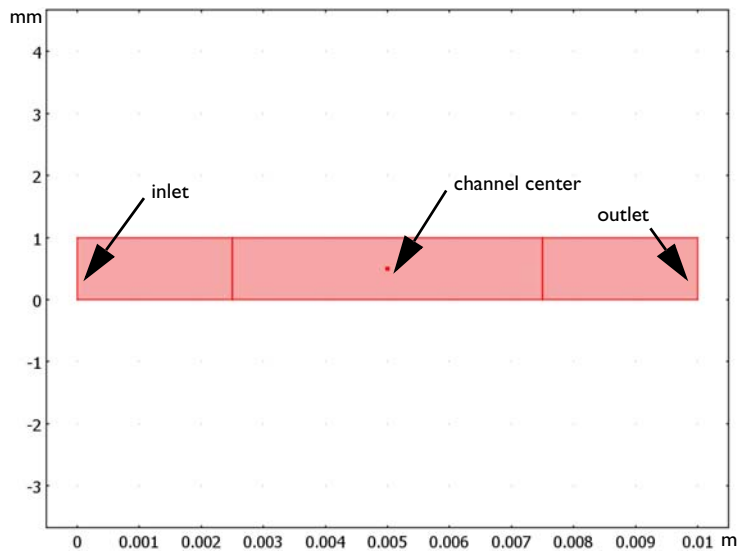
Topology optimization of the Navier-Stokes equations is encountered in different branches and applications, such as in the design of ventilation systems for cars and optimal reactors. A common technique applicable to such problems is to let the distribution of porous material vary continuously. In this model, the objective is to find the optimal distribution of a porous material in a microchannel such that the horizontal velocity component at the center of the channel is minimized.

The model is inspired by Ref. 1.

## *Model Definition*

---

The model geometry (Figure 11-1) consists of three regions: the inlet channel, the design domain, and the outlet channel. A prescribed pressure drop between the inlet and the outlet drives the flow.



*Figure 11-1: The model geometry.*

The fluid flow in the channel is described by the Navier-Stokes equations

$$\begin{aligned} \rho(\mathbf{u} \cdot \nabla)\mathbf{u} &= -\nabla p + \nabla \cdot \eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T) - \alpha(\gamma)\mathbf{u} \\ \nabla \cdot \mathbf{u} &= 0 \end{aligned} \quad (11-6)$$

where  $-\alpha(\gamma)\mathbf{u}$  is a force term in which the coefficient

$$\alpha(\gamma) \equiv \alpha_{\max} \frac{q(1-\gamma)}{q+\gamma} \quad (11-7)$$

characterizes the flow in a porous medium. In Equation 11-7,  $\alpha$  is interpreted as a continuous mapping determined by the function  $\gamma: \Omega \rightarrow [0, 1]$ , which in the limit of decreasing Darcy number and decreasing mesh size should be a discrete-valued function. When  $\gamma$  equals 1,  $\alpha$  is zero, corresponding to free flow. Conversely, for  $\gamma = 0$ ,  $\alpha = \alpha_{\max}$ , where  $\alpha_{\max}$  is related to the dimensionless Darcy number,  $\text{Da}$ , according to

$$\text{Da} = \frac{\eta}{\alpha_{\max} L^2} \quad (11-8)$$

The convergence of the optimization process depends on three important factors: the Darcy number, the mesh size, and the coefficient  $q$ . Rewriting Equation 11-7 it is easily seen that  $\alpha/\alpha_{\max} \rightarrow 1 - \gamma$  in the limit  $q \rightarrow \infty$ . In this limit,  $\gamma$  can be interpreted as the local porosity, ranging between 0 (filled) and 1 (open channel).

### *Results and Discussion*

---

Figure 11-2 displays the design variable,  $\gamma$ , which represents the distribution of porous material. As the plot shows,  $\gamma$  is either 0 or 1 in most of the domain, with a narrow transition zone in between. The width of this transition zone is mesh dependent; you can reduce it by decreasing the mesh-element size. Alternatively, decreasing the Darcy number also gives harder boundaries at the interface between porous and open domains.

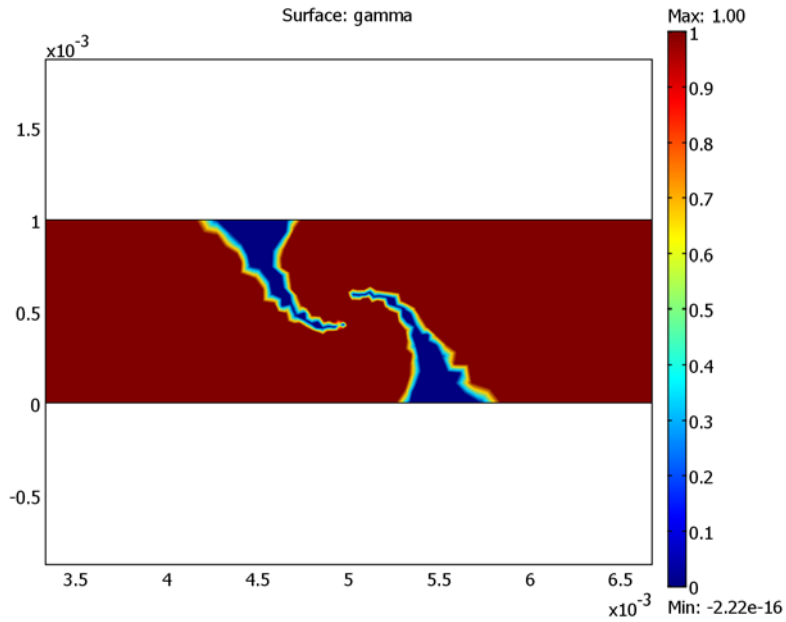


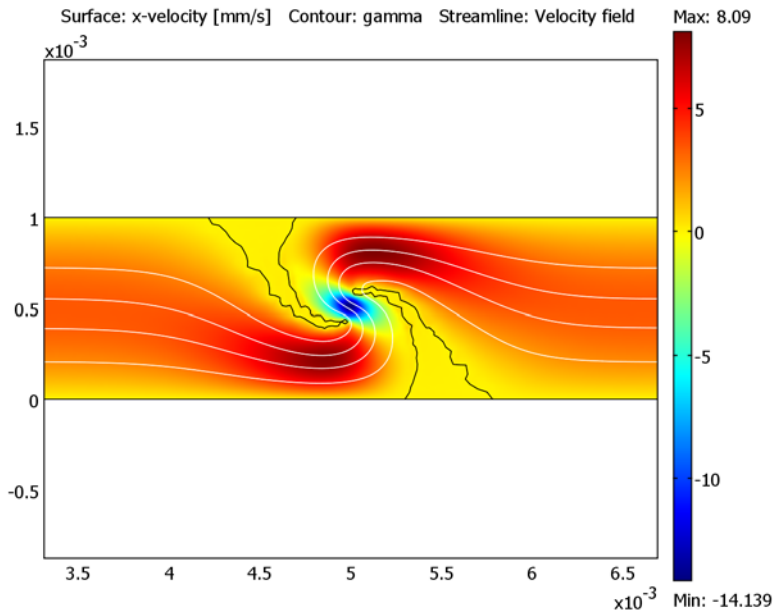
Figure 11-2: Distribution of porous material; the red areas represent open channel.

A question that naturally arises in this type of problems concerns uniqueness of the solution. In this case, there is at least one more solution that gives exactly the same result; because the channel has no upside or downside, a solution mirrored around the axis  $y = 0.5$  mm would give exactly the same flow.

Figure 11-3 contains a surface plot of the horizontal velocity component and a streamline plot of the velocity field resulting from the optimization process. In addition, the contour  $\gamma = 0.5$  indicates the border between the open channel and filling material. The plots reveal how the flow turns around, with a negative horizontal



velocity at the center of the channel. Note also that the  $x$ -velocity has a minimum of roughly  $-15$  mm/s at the design point.



*Figure 11-3: The horizontal velocity (surface plot) and velocity field (streamlines) after optimization. In addition, the contour  $\gamma = 0.5$  indicates the border between open channel and filling material.*

If you were to increase the streamline density in the above plot, some streamlines passing through the barriers would appear. This effect is due to a small amount of leakage, which can be reduced further by increasing the mesh resolution.

Figure 11-4 shows the pressure distribution, verifying the prescribed pressure drop through the channel length. The pressure drop is naturally concentrated to the region with porous material.

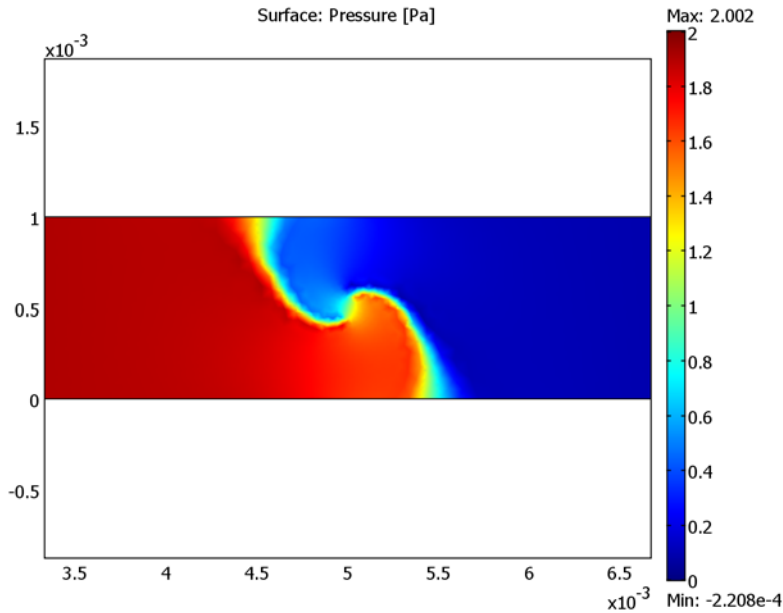


Figure 11-4: Pressure distribution in the channel, the pressure drop is concentrated to the porous domain.

### Reference

---

I. L. Højgaard Olesen, F. Okkels, and H. Bruus, “A high-level programming-language implementation of topology optimization applied to steady-state Navier-Stokes flow,” *Int. J. Num. Meth. Engrg*, vol. 65, pp. 975–1001, 2005.

### Modeling in COMSOL Multiphysics

---

This model combines the Optimization application mode and the Incompressible Navier-Stokes application mode. First, you calculate the solution for the flow in an empty channel (that is, with no porous material). You then use this solution as a starting point for the optimization. In each iteration, the software calculates a solution for the flow problem and feeds it to the optimization routine, which updates the design variables. If the specified convergence criterion is fulfilled, the solution process

terminates; otherwise the new design-variable values are used in the next calculation of the flow problem.

Setting up this kind of model with a general optimization routine would require quite a bit of work, but as you will discover, solving this problem with the built-in tools for optimization in COMSOL Multiphysics is easy.

---

**Model Library path:** COMSOL\_Multiphysics/Fluid\_Dynamics/reversed\_flow

---

### *Modeling Using the Graphical User Interface*

---

#### **MODEL NAVIGATOR**

- 1 In the **Model Navigator** begin by clicking the **Multiphysics** button.
- 2 Select **COMSOL Multiphysics>Fluid Dynamics>Incompressible Navier-Stokes**, then click **Add**.
- 3 Select **COMSOL Multiphysics>Optimization and Sensitivity>Optimization**, then click **Add**.
- 4 Click **OK** to close the **Model Navigator**.

#### **GEOMETRY MODELING**

- 1 Create three rectangles according to the following table. For each rectangle, Shift-click the **Rectangle/Square** button on the Draw toolbar; then specify width, height, and corner position; and finally click **OK**.

WIDTH	HEIGHT	CORNER
2.5e-3	1e-3	(0, 0)
5e-3	1e-3	(2.5e-3, 0)
2.5e-3	1e-3	(7.5e-3, 0)

- 2 Click the **Zoom Extents** button to fit the model geometry to the drawing area.
- 3 Draw a point at the center of the channel by shift-clicking the **Point** button and then entering 5e-3 and 0.5e-3 in the **x** and **y** edit fields, respectively.

The geometry should now look like that in Figure 11-1.

#### **CONSTANTS AND EXPRESSIONS**

- 1 From the **Options** menu, open the **Constants** dialog box.

2 Define constants according to the following table (the descriptions are optional):

NAME	EXPRESSION	DESCRIPTION
q	1	Optimization parameter
Da	1e-5	Darcy number
L	1[mm]	Inlet height
rho	1e3[kg/m^3]	Density of water
eta	1e-3[Pa*s]	Viscosity of water
u0	10[mm/s]	Flow velocity scale

The purpose of the constant u0 is to introduce a rough velocity scale that you can use to make the objective function dimensionless with a value of the order 1.

3 Click **OK** to close the dialog box.

4 Choose **Options>Expressions>Subdomain Expressions**.

5 Select Subdomain 2, then enter names and expressions according to the following table:

NAME	EXPRESSION
alpha_max	eta_ns / (Da*L^2)
alpha	alpha_max*q*(1-gamma) / (q+gamma)

6 Click **OK**.

## PHYSICS SETTINGS

### *Subdomain Settings—Incompressible Navier-Stokes*

1 From the **Model Tree** or the **Multiphysics** menu, select the **Incompressible Navier-Stokes (ns)** application mode.

2 From the **Physics** menu, open the **Subdomain Settings** dialog box.

3 Select all subdomains, and type rho in the **Density** edit field and eta in the **Dynamic viscosity** edit field.

4 Select Subdomain 2, and type -alpha\*u in the **Volume force, x dir.** edit field and -alpha\*v in the **Volume force, y dir.** edit field.

5 Click **OK**.

### *Boundary Conditions—Incompressible Navier-Stokes*

In the next steps, you specify the pressure drop along the channel length by prescribing the pressure at the inlet and at the outlet. The resulting pressure gradient drives the flow in the channel.

- 1 From the **Physics** menu, open the **Boundary Settings** dialog box.
- 2 Enter the boundary conditions according to the following table:

SETTINGS	BOUNDARY 1	BOUNDARY 10
Boundary type	Inlet	Outlet
Boundary condition	Pressure, no viscous stress	Pressure, no viscous stress
$P_0$	2	0

Keep all other boundaries at the default condition, which is the no-slip condition.

- 3 Click **OK**.

#### *Subdomain Settings—Optimization*

- 1 From the **Model Tree**, select **Optimization (opt)**.
- 2 Open the **Subdomain Settings** dialog box.  
Only the center part of the channel geometry is needed in the optimization, so you can deactivate the other subdomains.
- 3 Select Subdomains 1 and 3, and then clear the **Active in this domain** check box.
- 4 Select Subdomain 2 and go to the **Variables** page.
- 5 Edit the fields according to the following table:

VARIABLE	INIT	ELEMENT
gamma	1	Lagrange - Linear

This defines shape functions for **gamma**, which is the design variable used in the optimization. The initial value 1 corresponds to a channel free from porous material. In the next step, you constrain the design variable to the range [0, 1].

- 6 Click the **Pointwise Constraints** tab.
- 7 Type 0 in the **lb** field, **gamma** in the **Expression** field, and 1 in the **ub** field.
- 8 Click **OK**.

#### *Point Settings—Optimization*

- 1 Choose **Physics>Point Settings**.
- 2 Select Point 5 from the list. On the **Objective** page, type  $u/u_0$  in the  **$q_0$**  edit field.  
This defines the objective function to be proportional to the  $x$ -component of the velocity at the center and normalized by the velocity-scale constant  $u_0$ .
- 3 Click **OK**.

## MESH GENERATION

This model is naturally highly dependent on the mesh size. In this example, choose a dense mesh; note, however, that this mesh can be further improved.

- 1 From the **Mesh** menu, open the **Free Mesh Parameters** dialog box.
- 2 On the **Global** page, click the **Custom mesh size** button.
- 3 In the **Element growth rate** edit field, type 1.1.
- 4 Click the **Subdomain** tab. Select all three subdomains.
- 5 In the **Subdomain mesh parameters** area, select **Triangle** from the **Method** list.
- 6 Select Subdomain 2 only, then set the **Maximum element size** to 0.12e-3.
- 7 On the **Point** page, select Point 5 and set the **Maximum element size** to 0.03e-3.
- 8 Click the **Remesh** button to generate the mesh.
- 9 When the mesher has finished, click **OK** to close the **Free Mesh Parameters** dialog box.

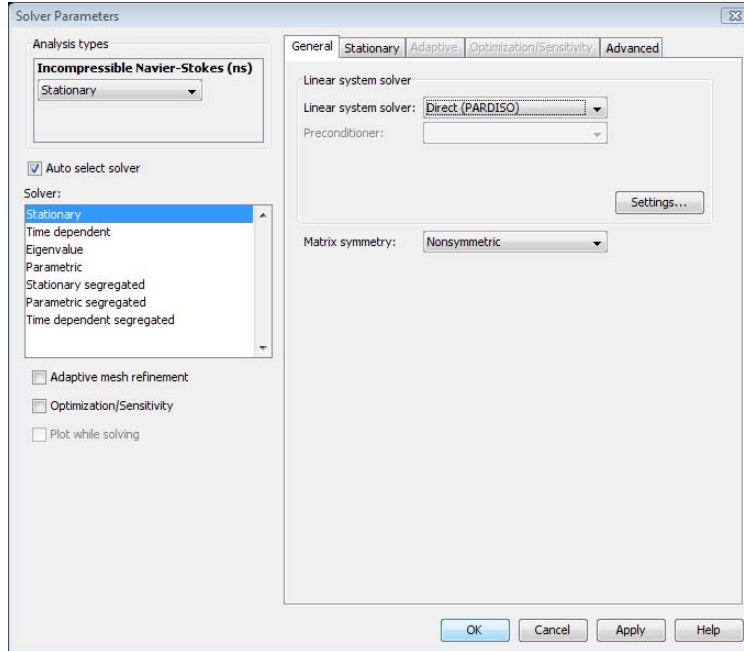
### *Computing the Solution—Static Flow*

---

First solve the model for an empty channel (that is, with no porous material) to check that the initial solution looks reasonable. Once you have established that this is the case, you can use this solution as a starting point for the optimization. Note that, as an alternative, it is possible to solve the full optimization problem in one go.

- 1 Click the **Solver Parameters** button on the Main toolbar.
- 2 From the **Linear system solver** list, select **Direct (PARDISO)**.

3 Make sure that **Optimization/Sensitivity** is not selected.



4 Click **OK** to close the **Solver Parameters** dialog box.

5 Click the **Solve** button to compute the solution.

The default plot shows the velocity field in the channel.

## COMPUTING THE SOLUTION—OPTIMIZATION

1 Open the **Solver Parameters** dialog box.

2 Select the **Optimization/Sensitivity** check box, then click the **Optimization/Sensitivity** tab.

3 From the **Analysis** list, select **Optimization**. Click **OK**.

4 Open the **Solver Manager**.

5 On the **Initial Value** page, click the **Current solution** button in the **Initial value** area, then click **OK**.

6 Click the **Solve** button on the Main toolbar to solve the model.

The optimization routine requires approximately 185 iterations and takes about 10 to 15 minutes depending on the computer.

## POSTPROCESSING AND VISUALIZATION

The default plot shows the velocity field. To evaluate the horizontal velocity in the center of the channel, follow these steps:

- 1 Choose **Postprocessing>Point Evaluation**.
- 2 Select Point 5 from the list.
- 3 In the **Expression** edit field, enter  $u$ .
- 4 Click **OK**.

The value appears in the lower part of the interface and should be about  $-0.013$  m/s.

To plot the design variable,  $\gamma$ , as seen in Figure 11-2, do the following:

- 1 Click the **Plot Parameters** button on the Main toolbar.
- 2 Click the **Surface** tab. Type  $\gamma$  in the **Expression** edit field on the **Surface Data** page or select **Optimization (opt)> $\gamma$**  from the **Predefined quantities** list.
- 3 Click **Apply** to generate the plot.
- 4 Click the **Zoom In** button on the Main toolbar twice to zoom in on the geometry.

To generate the pressure plot in Figure 11-4 proceed as follows:

- 1 Still on the **Surface Data** page, change the text in the **Expression** edit field to  $p$ .
- 2 Click **Apply**.

Finally, generate Figure 11-3 with the following instructions:

- 1 Change the text in the **Expression** edit field to  $u$  (or select **Incompressible Navier-Stokes (ns)>x-velocity** from the **Predefined quantities** list).
- 2 From the **Unit** list, select **mm/s**.
- 3 Click the **Contour** tab and select the **Contour plot** check box.
- 4 On the **Contour Data** page, type  $\gamma$  in the **Expression** edit field.
- 5 In the **Contour levels** area, click the **Vector with isolevels** option button and then type  $0.5$  in the associated edit field.
- 6 In the **Contour color** area, click first the **Uniform color** option button and then the **Color** button.
- 7 In the **Contour Color** dialog box select black, then click **OK**.
- 8 Clear the **Color scale** check box.
- 9 Click the **Streamline** tab and select the **Streamline plot** check box.
- 10 Leave the default selection (**Velocity field**) on the **Streamline Data** page.



- 11** From the **Streamline plot type** list, select **Magnitude controlled**.
- 12** On the **Density** page, set the **Density** to 10.
- 13** On the **Line Color** tab, click the **Color** button.
- 14** In the **Streamline Color** dialog box select white, then click **OK**.
- 15** In the **Plot Parameters** dialog box, click **OK** to close the dialog box and generate the plot.



## Global Equations and ODEs

This chapter describes the use of COMSOL Multiphysics to solve global equations such as ODEs, algebraic equations, and transcendental equations. You can solve such equations separately or coupled to a full finite-element model. Quick examples show how to specify and solve these types of equations.

# Solving ODEs and Global Equations

## *Adding ODEs and Other Global Equations*

---

When working on complex models, you frequently need to introduce single named degrees of freedom, or *states*, which are not logically related to any particular point in space, and their corresponding equations. In particular, such situations arise when modeling physics in interaction with an external system, for example, a controller or an electrical circuit built from standard components. It is often possible to describe such systems by a system of ordinary differential equations, ODEs, with a limited number of degrees of freedom.

The **Global Equations** dialog box is designed for implementing this type of external equation, tightly coupled to a PDE model in a physical domain. You can use it for ODEs, differential algebraic equations, purely algebraic equations and conditions, and transcendental equations, or to add degrees of freedom to a PDE using the introduced states. Possible uses include:

- Controllers (see “Process Control Using a PID Controller” on page 337 in the *COMSOL Multiphysics Model Library*)
- Rigid-body mechanics (see “Terminal Falling Velocity of a Sand Grain” on page 226 in the *COMSOL Multiphysics Model Library*)
- Nonlinear eigenvalue problems (see “Example—Eigenmode Analysis of an L-Shaped Membrane with Rounded Corner” on page 487)
- Continuation
- Integral constraints (see “Example—Eigenmode Analysis of an L-Shaped Membrane with Rounded Corner” on page 487 of this book and “The Two-Term Boltzmann Equation” on page 188 in the *COMSOL Multiphysics Model Library*)
- Augmented or generalized equations (see “Stress-Optical Effects with Generalized Plane Strain” on page 461 in the *Structural Mechanics Module Model Library*, also available in the *RF Module Model Library*)

## *The Global Equations Dialog Box*

---

To add a space-independent equation such as an ODE, choose **Global Equations** from the **Physics** menu. On the **States** page, each row corresponds to a named state, that is, a single degree of freedom. Fill in the name of the state variable in the first column

(**Name (u)**). This also defines time-derivative variables. If a state variable is called  $u$ , its first and second time derivatives are  $u_t$  and  $u_{tt}$ , respectively. These variables become available in all domains in all geometries. Therefore the names must be unique.

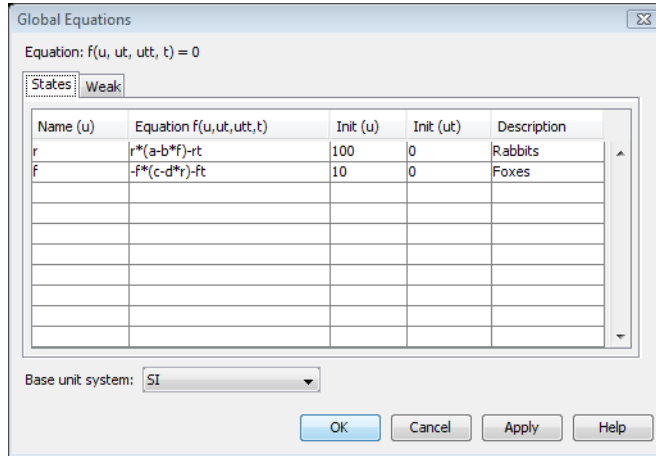


Figure 12-1: The Global Equations dialog box, States page. The ODEs in the dialog box are the Lotka-Volterra or predator-prey equations.

Use the **Equation** column to specify an expression that is set equal to zero and added to the global system of equations. You can use all state variables and their time derivatives as well as any constants, global expression variables, and integration-coupling variables with a global destination. Setting an equation for a state is optional. The default value of 0 means that COMSOL Multiphysics does not add any additional condition to the model.

If the time derivative of a state variable appears somewhere in the model during a time-dependent solution, the state variable needs an initial condition. Models that contain second time derivatives also require an initial value for the first time derivatives of the state variables. You set these conditions in the third (**Init (u)**) and fourth (**Init (ut)**) columns. The last column, **Description**, gives you the option to enter comments about the state or the equation.

### *Solving ODEs—the Lotka-Volterra Equations*

As an example, the ODEs in Figure 12-1 are the Lotka-Volterra equations:

$$\begin{aligned}\dot{r} &= ar - brf \\ \dot{f} &= -cf + drf\end{aligned}$$

where  $r$  is the rabbit population, and  $f$  is the population of foxes.

Another option is to enter equations in the weak form on the **Weak** page. This is convenient in advanced modeling because it gives you control over the test variables multiplying the equations. Wherever a test function of a state variable appears (in the **Global Equations** dialog box or elsewhere in the model), whatever it multiplies ends up in the same equation in the discrete system. On the **Weak** page you can also use test functions of integration-coupling variables. This way you can keep the definition of a controller and its influence on a PDE in the same dialog box. Note that you can have zero or more weak expressions, regardless of the number of states.

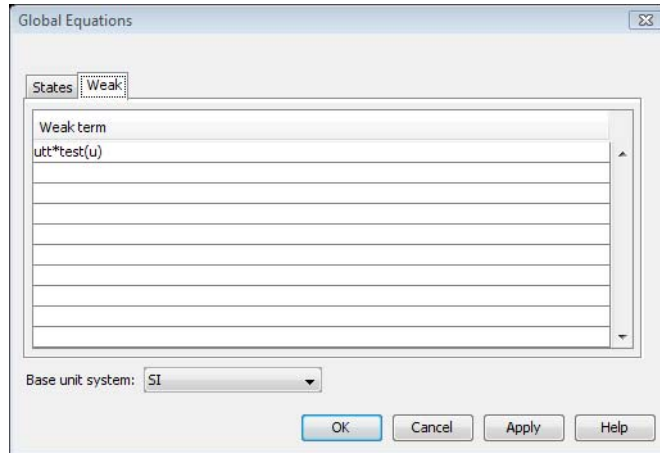


Figure 12-2: The Global Equations dialog box, entering an ODE in the weak form.

To postprocess the solution to an ODE, type the name of the state variable in the **Expression** edit field in most postprocessing dialog boxes, for example, on the **Point** page in the **Domain Plot Parameters** dialog box for a plot of the time evolution. The state variables are available globally so you can select any point in the geometry.

### *Solving Algebraic and Transcendental Equations*

As an example of an algebraic equation, consider the following:

$$f(u) = u^3 + u - 2$$

This equation has a single root at  $u = 1$ . To enter it into the **Global Equations** dialog box, type  $u$  in the **Name (u)** column and  $u^3+u-2$  in the **Equation** column (both entries on the same row). Then click the **Solve** button on the Main toolbar to compute the solution. To display the solution in the message log, choose **Postprocessing>Data Display>Global** and type  $u$  in the **Expression** edit field in the **Global Data Display** dialog box. Click **Apply**, and **Value: 1, Expression: u** appears in the message log.

As an example of a transcendental equation, consider the following:

$$f(u) = e^{-u} - u$$

A root to this equation is approximately 0.56714. To enter it into the **Global Equations** dialog box, type  $u$  in the **Name (u)** column and  $\exp(-u) - u$  in the **Equation** column (both entries on the same row). Then click the **Solve** button on the Main toolbar to compute the solution. To display the solution in the message log, choose **Postprocessing>Data Display>Global** and type  $u$  in the **Expression** edit field in the **Global Data Display** dialog box. Click **Apply**, and **Value: 0.567143, Expression: u** appears in the message log.

### *Adding an ODE to a Boundary*

---

Sometimes a boundary equation is coupled to a PDE in an adjacent subdomain. For example, adding an ODE for  $v$  on a boundary such that

$$\frac{\partial v}{\partial t} = u$$

where  $u$  is the dependent variable in an adjacent subdomain. To implement an ODE on a boundary, do the following:

- 1 Add a Weak Form, Boundary application mode.
- 2 Go to the **Physics** menu and choose **Boundary Settings**. In the **Boundary Settings** dialog box, select the boundary or boundaries where you want to defined the ODE.
- 3 Click the **Weak** tab and type  $v\_test*u$  in the **weak** edit field and  $v\_test*v\_time$  in the **dweak** edit field. This is the weak formulation of the ODE.
- 4 Select the remaining boundaries where there is no ODE and then clear the **Active in this domain** check box to exclude these boundaries.
- 5 When done, click **OK**.

Also keep in mind that there must be no constraint (Dirichlet condition) for  $u$  on the boundaries where you defined the ODE.





## The Weak Form

This chapter explains weak form modeling, its purpose, and how to use it. The weak form in COMSOL Multiphysics is a particular way of specifying a model, one in which you use a more general syntax to describe the problem. The discussion concludes with a section giving some theoretical background.

Do not be misled by the term “weak:” the weak form is very powerful and flexible. We have borrowed the term *weak form* from mathematics, but in this context it has a slightly different meaning; this implementation incorporates features in addition to those defined in the mathematical weak form. Moreover, knowledge of the mathematical weak form is not prerequisite to using the COMSOL Multiphysics implementation.

The distinguishing features of the weak form in COMSOL Multiphysics are that it makes it possible to:

- Solve strongly nonlinear and nonlocal models that need an exact Jacobian for convergence.
- Enter certain equations which can be derived from an energy principle in a very compact and convenient form. Such equations, for example, arise in structural mechanics.

- Add and modify nonstandard constraints, such as various contact and friction models.
- Build models with PDEs on boundaries, edges, and points.
- Use the test operator to conveniently work with problems in variational calculus and parametric optimization.

COMSOL Multiphysics provides two different ways of modeling with the weak form:

- Weak form application modes
- Weak form contributions

In addition, you can add *weak constraints*, which, for example, provide accurate fluxes and reaction forces.

There are many additional benefits of the weak form, and we mention some of them in the context of specific models in other books in this documentation set.

# Weak Form Modeling

The *weak form application modes* enable all of the weak form features outlined above. In particular, you need to use these application modes for models with separate PDEs on boundaries, edges, or points. However, modeling with the weak form does not always require using the weak form application modes; sometimes adding *weak form contributions* is sufficient, as described below.

By default, all models are converted to the *weak solution form* before solving. Although this feature is a part of the solution technique rather than part of the modeling process, it nonetheless belongs in this discussion because it is based on the weak form implementation. Using the weak solution form gives you the most important benefit of the weak form—the exact Jacobian necessary for fast convergence of strongly nonlinear problems—while you can still do the modeling and view the equations in the *equation form* of your choice.

The discussion in this chapter reviews the easiest ways to work with the weak form before going on to describe the weak form application modes.

## *Adding Weak Form Contributions*

---

To add weak contributions to a model you are describing in a physics mode, follow these steps:

- 1 In the **Physics** menu, point to **Equation System** and then select **Subdomain Settings**, **Boundaries Settings**, **Edge Settings**, or **Points settings** depending on where you need to add the contributions.
- 2 In the corresponding dialog box, click the **Weak** tab.
- 3 Enter the weak contribution in the **weak** or **dweak** edit fields.

To add weak contributions to a model you are describing in a coefficient form or general form PDE mode, use the **Subdomain Settings** dialog box and other settings directly available in the **Physics** menu.

### **ADDING EDGE AND POINT SOURCES AND CONSTRAINTS**

Using weak form contributions, you can also add sources and constraints on edges and points. The section “Specifying Point and Edge Settings” on page 254 in the *COMSOL Multiphysics User’s Guide* describes the procedure.

## **ADDING WEAK CONSTRAINTS**

The weak form can assist in making accurate flux or reaction-force computations. The section “Using Weak Constraints” on page 358 describes how to use weak constraints for this purpose.

## **VARIATIONAL CALCULUS**

Use the test operator to conveniently work with problems in variational calculus and optimization theory. See the minimal-surface example on page 354 for an example of this technique.

### *Modeling with PDEs on Boundaries, Edges, and Points*

---

The weak form makes it possible to do PDE modeling on boundaries, edges, and points, thus opening up a wider class of models than the other application modes. This is the only type of modeling where you need the weak form application modes.

Using weak form application modes is also a smart way of handling thin layers; COMSOL Multiphysics then solves the problem by modeling rather than meshing. This approach reduces the solution time, and is sometimes what makes a solution possible at all.

The model “Transport and Adsorption” on page 32 of the *COMSOL Multiphysics Model Library* shows how to use the weak form boundary mode to model a thin adsorption layer with diffusion as a PDE on the boundary of a convection-diffusion problem. Another model using tangential derivative variables is “Shell Diffusion” on page 156 of the *COMSOL Multiphysics Model Library*.

### *Using Weak Coefficients or the Weak Modes*

---

## **USING THE WEAK SOLUTION FORM**

By using the weak solution form (the default solution form) you can model with one of the strong forms of the PDE using, for example, the PDE application modes, yet still take advantage of some of the abilities of the weak form. COMSOL Multiphysics converts all your equations to the weak form before solving the problem.

## WEAK FORM VS. STRONG FORMS

Some features in COMSOL Multiphysics are available only in either strong or weak form. When a feature is available in the weak form but not in a strong form, you can always

- Display and modify the equations in the weak form using the weak equation system form
- Convert to the weak form when solving using the weak solution form (this is the default behavior)

When a feature is available only in a strong form, however, you must model using this form.

### *Strong Form Features*

- When using residual method *Coefficient*, the adaptive mesh refinement considers only the strong form coefficients when computing the error estimators. The default residual method *Weak* accounts for all contributions, on weak and strong form.
- The equation variables and coupled boundary equation variables are available only for the strong forms.

### *Weak Form Features*

The weak form automatically computes the Jacobian contributions for all variables, including coupling variables. The strong forms only consider Jacobian contributions from coefficients. Thus, to get an accurate Jacobian when dealing with for example coupling variables and mixed time-space derivatives you might have to use the weak solution form.

# Example—Variational Calculus

Variational calculus is a branch of mathematics where you want to find the function that extremizes some functional and often also determine the extremal value of the functional in question in the process. According to a fundamental result in the calculus of variations, the integral

$$\int_{\Omega} F(x, u, \nabla u) d\Omega$$

has a stationary value if the *Euler-Lagrange equation*

$$\frac{\partial F}{\partial u} - \nabla \cdot \frac{\partial F}{\partial \nabla u} = 0$$

is satisfied. Use the test operator to conveniently handle certain problems from variational calculus as it implements the Euler-Lagrange equation. See “The test Operator” on page 164 of the *COMSOL Multiphysics User’s Guide* for its definition.

## *Model Definition*

---

The area of the surface  $\{z = u(x, y) \mid (x, y) \in \Omega\}$  is equal to

$$A[u] = \int_{\Omega} \sqrt{1 + |\nabla u|^2} d\Omega$$

Given the values of the function  $u$  on the boundary  $\partial\Omega$ , the task is to determine  $u$  inside  $\Omega$  such that the area is minimized. Solving this minimal-surface problem is related to finding the shape of a soap film with a prescribed boundary.

In this example, let  $\Omega = \{(x, y) \mid x^2 + y^2 \leq 1\}$  with  $u = x^2$  on the boundary  $\partial\Omega$ . Using the Euler-Lagrange equation, you can derive the equation for the minimal-surface problem as:

$$-\nabla \cdot \left( \frac{1}{\sqrt{1 + |\nabla u|^2}} \nabla u \right) = 0$$

Simplify the problem by using the test operator to let COMSOL Multiphysics derive the minimal surface problem automatically. Typing in

$$\int_{\Omega} \text{test}(\sqrt{1 + |\nabla u|^2}) d\Omega$$

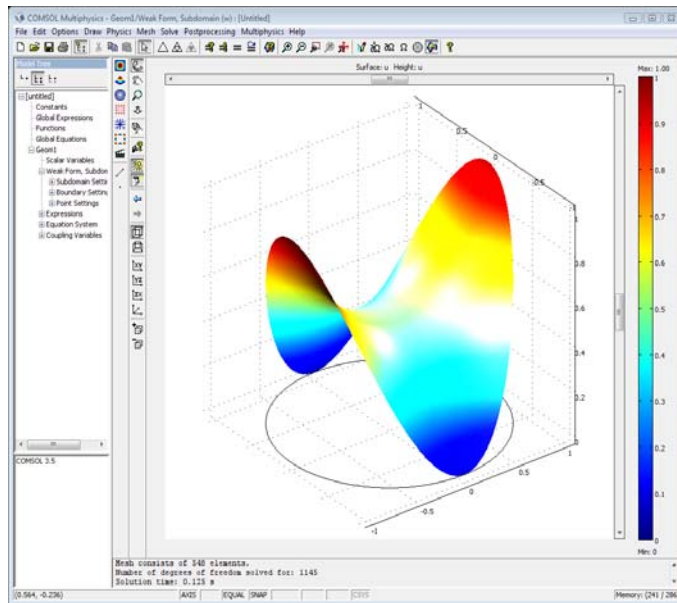
is equivalent to typing in

$$\int_{\Omega} \left( \text{test}(u) \frac{\partial}{\partial u} \sqrt{1 + |\nabla u|^2} + \text{test}(\nabla u) \frac{\partial}{\partial \nabla u} \sqrt{1 + |\nabla u|^2} \right) d\Omega$$

which is the weak form of the Euler-Lagrange equation.

## Results

The shape of the solution illustrates how a soap film would be suspended in space, as constrained by the boundary.



## Modeling in COMSOL Multiphysics

Solve the problem directly in the weak form, using the stationary solver.

---

**Model Library path:** COMSOL\_Multiphysics/Benchmarks/minimal\_surface

---

## *Modeling Using the Graphical User Interface*

---

### MODEL NAVIGATOR

- 1 Select **2D** from the **Space dimension** list.
- 2 In the application mode list, open **COMSOL Multiphysics>PDE Modes** and then select **Weak Form, Subdomain**. Make sure **Lagrange - Quadratic** elements are selected.
- 3 Click **OK**.

### GEOMETRY MODELING

The problem domain is a unit circle. Draw a circle centered at (0, 0) with a radius of 1.

### PHYSICS SETTINGS

#### *Boundary Conditions*

- 1 From the **Physics** menu, choose **Boundary Settings**.
- 2 Enter the following boundary coefficients:

SETTINGS	ALL BOUNDARIES
weak	0
dweak	0
constr	$x^2 - u$

- 3 Click **OK**.

#### *Subdomain Settings*

- 1 From the **Physics** menu, open the **Subdomain Settings** dialog box and enter the PDE coefficients:

SETTINGS	SUBDOMAIN I
weak	$\text{test}(\sqrt{1+u_x^2+u_y^2})$
dweak	0
constr	0

- 2 Click **OK**.



## MESH GENERATION

Click the **Initialize Mesh** button to create a mesh.

## COMPUTING THE SOLUTION

Click the **Solve** button on the Main toolbar to solve the problem.

## POSTPROCESSING AND VISUALIZATION

Click the **3D Surface Plot** button and the **Scene Light** button to create an illuminated 3D plot.

### *Modeling Using the Programming Language*

---

Run the model from COMSOL Script or MATLAB:

- 1 Create the geometry and a standard mesh:

```
clear fem
fem.geom = circ2(0,0,1);
fem.mesh = meshinit(fem);
```

- 2 Set up the minimization problem that characterizes the minimal surface. Use second order Lagrange elements:

```
fem.shape = 2;
fem.sshape = 2;
fem.bnd.constr = 'x^2-u';
fem.equ.weak = 'test(sqrt(1+ux^2+uy^2))';
```

- 3 Create the extended mesh, solve the nonlinear problem, and plot the solution:

```
fem.xmesh = meshextend(fem);
fem.sol = femstatic(fem);
postsurf(fem, 'u', 'u');
```

- 4 Compute the area of the minimal surface:

```
postint(fem, 'sqrt(1+ux^2+uy^2)')
```

# Using Weak Constraints

The *weak constraints* feature in COMSOL Multiphysics implements constraints by using finite elements on the constrained domain for the Lagrange multipliers, and by solving for the Lagrange multipliers along with the original problem. The weak constraints have a number of distinct advantages. They can:

- Provide very accurate flux computations—the Lagrange multipliers along the constraints optimally balance the finite element projection of the applied loads (reaction forces). See “Computing Accurate Fluxes” on page 266 in the *COMSOL Multiphysics User’s Guide* for more information and an example. For computing integrals of reaction forces or fluxes during postprocess, the reaction force operator (`reactf`) is a better alternative than weak constraints. See the section “The `reactf` operator” on page 168 of the *COMSOL Multiphysics User’s Guide*. The reaction force operator does not have the drawbacks discussed below.
- Handle nonlinear constraints—the nonlinear solver does not store the Lagrange multipliers arising from standard constraints from one step to the next, which affects the convergence when constraints are nonlinear. Weak constraints can handle general nonlinearities because the Lagrange multipliers are updated as a part of the solution vector and give correct contributions to the stiffness matrix.
- Implement constraints including derivatives—constraints only on the tangential component of the derivative work when using standard constraints, whereas you must use weak constraint to be able to handle nontangential constraints. Note that boundary conditions involving the normal component of the derivative can almost always be reformulated as a Neumann condition, which is usually preferable.

Weak constraints also come with the following drawbacks:

- Because extra unknowns are introduced for the Lagrange multipliers the problem size increases compared to when eliminating the constraints.
- The formulation normally implies that a saddle-point problem is introduced with zeros on the main diagonal of the Jacobian of the discretized equations. This class of problems is often more difficult to solve. Because iterative methods for linear systems are sensitive to the eigenvalue distribution of the system matrix, a weak constraint formulation can be much more difficult to solve than the corresponding pointwise constraint formulation.
- Discontinuous constraints result in (theoretically) infinite Lagrange multipliers. In practice you get large oscillations.

## Specifying Weak Constraints

---

In the graphical user interface you can add weak constraints using the **Weak constraints** application mode property. The following procedure applies to the physics modes:

- 1 From the **Physics** menu select **Properties**.
- 2 In the **Application Mode Properties** dialog box, select **On** from the **Weak constraints** list.
- 3 Select **Ideal** or **Non-ideal** from the **Constraint type** list (see “Ideal vs. Non-Ideal Constraints” on page 359).
- 4 Click **OK**.

The variable names in the application modes using weak constraints, such as  $1m1$ , correspond to the Lagrange multipliers. The weak constraints supersede the original constraints and implement them using a weak formulation. By default, weak constraints are active on all boundaries. To control the use of weak constraints on individual boundaries, open the **Boundary Settings** dialog box and click the **Weak Constr.** tab. On this tab, which appears when you add weak constraints, you can select boundaries and then select or clear the **Use weak constraints** check box to include or turn off the weak constraints for the selected boundaries. You can also set element type, shape function, and initial values for the weak constraint variables (Lagrange multipliers).

### IDEAL VS. NON-IDEAL CONSTRAINTS

Ideal weak constraints in theory implement the same boundary conditions as the standard, pointwise, constraints. In some cases, the result may differ slightly, but this is only due to the different discretization. Using non-ideal constraints, on the other hand, modifies the way boundary conditions are interpreted. Switching from pointwise constraints to non-ideal weak constraints therefore can modify the *physics* of the model, while using ideal weak constraints only affects the *numerics*.

The difference between ideal and non-ideal constraints is the way the Lagrange multipliers—which can be interpreted as generalized reaction forces—are applied. In an ideal constraint, the Lagrange multipliers are applied symmetrically on all dependent variables involved in the constraint so as to keep symmetric problems symmetric. In a non-ideal constraint, the reaction forces are applied only on the application mode’s dependent variable at the boundary where the constraint is specified.

Imagine a model with two application modes  $A$  and  $B$ , with dependent variables  $u$  and  $v$ , respectively. If application mode  $A$  specifies a standard constraint as  $R = 2u - 3v = 0$ , COMSOL Multiphysics implements the ideal weak boundary constraint by adding the weak term

$$\int_B \hat{\mu} \cdot (2u - 3v) ds + \int_B \mu \cdot (2\hat{u} - 3\hat{v}) ds$$

where  $\mu$  and  $\hat{\mu}$  are the Lagrange multiplier and corresponding test function, and  $\hat{u}$  and  $\hat{v}$  are the test functions corresponding to  $u$  and  $v$ , respectively. Since  $\mu$  multiplies both test functions in the second integral, both  $u$  and  $v$  are affected by the constraint, which is therefore *bidirectional*.

The corresponding non-ideal weak constraint adds the weak form contribution

$$\int_B \hat{\mu} \cdot (2u - 3v) ds + \int_B \mu \cdot \hat{u} ds$$

Here, the Lagrange multiplier  $\mu$  only multiplies the test function  $\hat{u}$ . All reaction forces therefore apply only to  $u$ , while  $v$  is left unaffected by the constraint. This makes the non-ideal constraint *unidirectional*. Note also that the non-ideal constraint does not consider the factor 2 in front of  $u$  when applying the forces. Even if the constraint would contain spatial derivatives, the Lagrange multiplier would still be applied only directly on the dependent variable itself.

For more information on the effects of ideal and non-ideal constraints, see “Example—Coupling Variables and Boundary Constraints” on page 362. For a model that uses ideal weak constraints for accurate outward normal current densities, see “Semiconductor Diode” on page 499 of the *COMSOL Multiphysics Model Library*. For a model that uses non-ideal weak constraints to get the correct reaction forces, see “Sloshing Tank” on page 265 of the *COMSOL Multiphysics Model Library*.

### *Weak Constraints in Assemblies*

---

When using assemblies, the application modes add a constraint on the dependent variables on the destination domains of the identity pairs. If you use weak constraints, the application modes also turn these constraints into weak constraints. Unlike other physics settings on pairs you do not control the use of weak constraints on pairs on the **Pair** tab in the **Boundary Settings** dialog box. If you would like to change the settings for the weak constraints on the destination boundaries of a pair go to the **Boundaries**

tab and change the settings there. See the section “Specifying Physics Settings on Pairs” on page 423 for general information about physics settings on pairs.

The constraints on pairs, which makes sure that the dependent variables are continuous across the pair boundary, are always implemented as ideal constraints. This because the non-ideal constraints do not conserve the flux from the source side to the destination side of the pair.

### *Limitations of Weak Constraints*

---

Weak constraints are more difficult to use than conventional pointwise constraints.

Consider the following tips when using them:

- Pointwise and weak constraints on the same set of variables on adjacent boundaries (boundaries that share common node points in the mesh) do not work. This means that if you must constrain all boundaries on a solid and want to use a weak constraint on one boundary segment (one face), you must use the weak constraint on the entire boundary of the solid (if the boundary is connected).
- You must always have a nontrivial constraint (that is, a Dirichlet condition) on the boundaries where you enable the weak constraint for the constraint to take effect.
- Problems with the scaling of linear systems sometimes arise in conjunction with weak constraints. The Lagrange multiplier variables always have a different unit than the main system variables, and can therefore be of a completely different order of magnitude. Usually the automatic variable scaling in the solvers is sufficient, but there are cases when you may want to consider manual scaling.
- Always use the same shape function type for the weak constraint as for the variables that you constrain, possibly with lower-order elements for the weak constraint. In some cases, for example when constraining derivatives, the system of equations may become singular. The reason is usually that there are redundant Lagrange multiplier degrees of freedom in the model. Try to lower the order of the Lagrange multiplier variables, or use constraints on the Lagrange multiplier to remove some degrees of freedom.
- If you want to use iterative methods for the linear system and if the weak constraint formulation gives zero's on the diagonal of the system matrix, the SOR class of preconditioners and smoothers will not work. Instead try use the Vanka algorithm with the Lagrange multipliers as the Vanka variables, or use the incomplete LU factorization algorithm.

# Example—Coupling Variables and Boundary Constraints

## *Boundary Constraints in a Heat Transfer Model*

---

This example describes the difference between different boundary constraints when using a coupling variable to map a function of a solution from one boundary to another.

Consider the following example:

### **MODEL NAVIGATOR**

- 1 In the **Model Navigator**, select **2D** in the **Space Dimension** list.
- 2 In the list of application modes, select **COMSOL Multiphysics>Heat Transfer>Conduction>Steady-state analysis**.

### **GEOMETRY MODELING**

- 1 Draw a rectangle with diagonal corners at (0, 0) and (2, 1).
- 2 Draw a circle with center at (1, 0.5) and radius 0.1.
- 3 Subtract the circle from the rectangle to create a hole: Press Ctrl+A to select both objects and click the **Difference** button.

### **PHYSICS SETTINGS**

#### *Boundary Conditions*

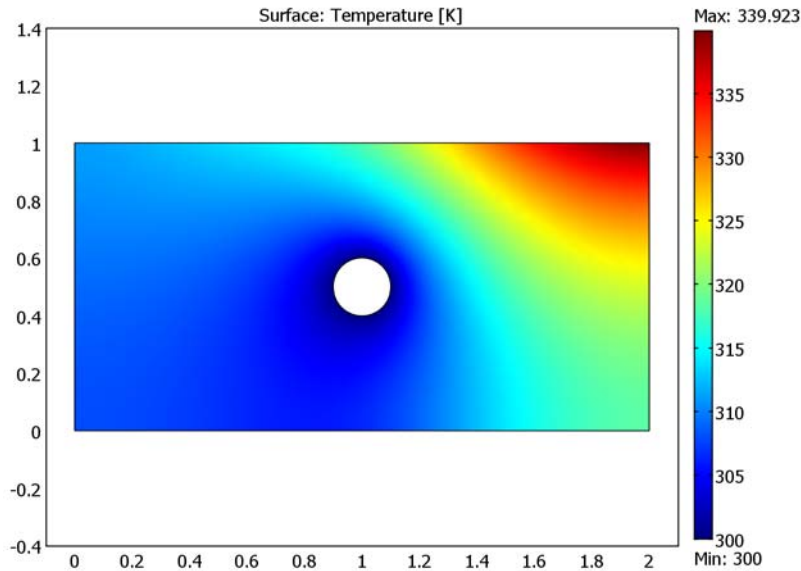
- 1 From the **Physics** menu, open the **Boundary Settings** dialog box.
- 2 Select the boundaries of the circle (Boundaries 5–8) and select **Temperature** in the **Boundary condition** list. Type 300 in the **Temperature** edit field to specify the value of the fixed temperature (in kelvin). The circle then acts as a heat sink.
- 3 Select Boundary 3 (the top boundary) and then select **Heat flux** in the **Boundary condition** list. Type  $10000 \cdot x$  in the **Inward heat flux** edit field. This creates a heat flux of 0 to 20,000 W/m<sup>2</sup>, which increases linearly from left to right.
- 4 The other boundaries are thermally insulated, which is the default setting. Click **OK** to close the **Boundary Settings** dialog box.

### Subdomain Settings

There are no changes to the settings in the **Subdomain Settings** dialog box (the default material properties are for copper).

### COMPUTING THE SOLUTION

Click the **Solve** button (=) on the Main toolbar (or select **Solve Problem** from the **Solve** menu) to solve the problem.



The maximum temperature, about 340 K, appears in the upper-right corner.

Now, what if you want to set the temperature along the left boundary of the rectangle equal to the average temperature of the right boundary? To do so, you can use an integration coupling variable, which computes the average temperature of the right boundary and stores the value in a variable.

### OPTIONS AND SETTINGS

- 1 From the **Options** menu, choose **Integration Coupling Variables>Boundary Variables**.
- 2 Select Boundary 4 (the rightmost boundary). Define an integration coupling variable  $T_{int}$  for the integrated temperature in the top row by typing  $T_{int}$  in the **Name** column and  $T$  in the **Expression** column. Also, define a second variable on the next row with the name  $b_{len}$  and an expression (integrand) that is 1. The variable

$T_{int}$  stores the value of the integral of  $T$  along the boundary, and the variable  $b_{len}$  stores the value of the length of the boundary. The average temperature then becomes  $T_{int}/b_{len}$ .

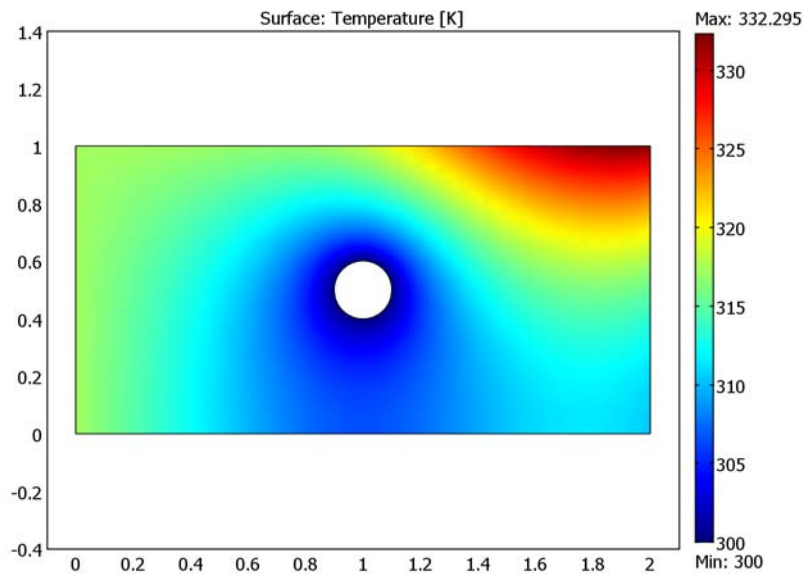
- 3 Click **OK** to close the **Boundary Integration Variables** dialog box.

#### **BOUNDARY CONDITIONS**

- 4 From the **Physics** menu, open **Boundary Settings**.
- 5 Select Boundary 1 (the leftmost boundary) and then select **Temperature** from the **Boundary condition** list.
- 6 Type  $T_{int}/b_{len}$  in the **Temperature** edit field.
- 7 Click **OK** to close the **Boundary Settings** dialog box.

#### **COMPUTING THE SOLUTION**

Click the **Solve** button on the Main toolbar.



The maximum temperature is about 332 K close to, but not exactly in, the upper-right corner. The default type of boundary constraint in COMSOL Multiphysics is an *ideal boundary constraint*, which is a bidirectional constraint. COMSOL Multiphysics supports two different kinds of ideal boundary constraints: pointwise and weak. Now



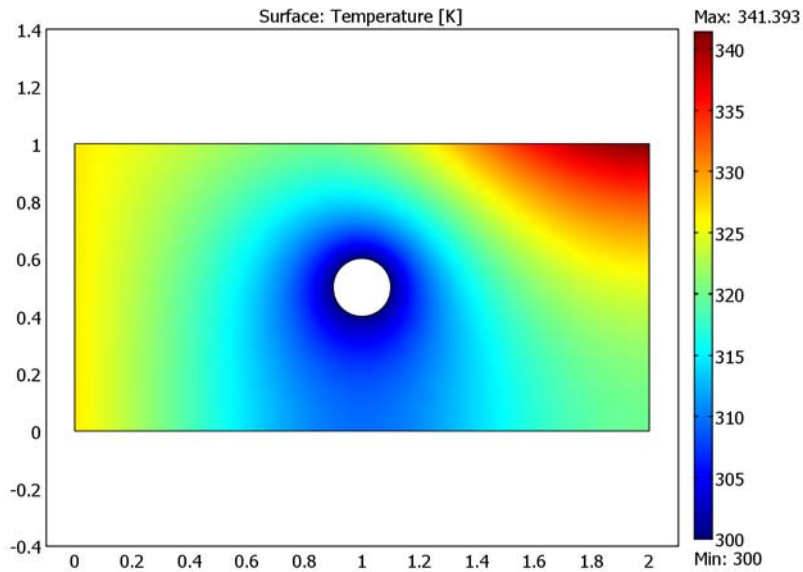
switch to a unidirectional, or non-ideal, boundary constraint, and solve the modified problem.

### PHYSICS SETTINGS

- 1 From the **Physics** menu, select **Properties**.
- 2 Select **On** from the **Weak constraints** list and **Non-ideal** from the **Constraint type** list to change from no weak constraints (equal to pointwise ideal constraint) to non-ideal weak constraints.
- 3 Click **OK** to close the **Application Mode Properties** dialog box.

### COMPUTING THE SOLUTION

Click the **Solve** button on the Main toolbar to solve the problem with the modified constraints.



This time the maximum temperature is about 341 K, in the exact upper-right corner.

### POSTPROCESSING

Do the following steps if you want to check that this is the average temperature:

- 1 From the **Postprocessing** menu, select **Data Display>Global**.

2 For the **Expression to evaluate**, type `Tint/blen`. Click **OK**.

This displays the average temperature value, about 327 K, in the message log. To compare this with the values in the plot, just click close to the left boundary. This displays the solution at the grid locations of the background grid, which you can see in the Draw mode. The same value appears for all points on the left boundary.

The subsequent sections explain the differences between the different types of boundary constraints.

### *No Weak Constraints*

---

The pointwise ideal boundary constraint is what you get if you select **Off** from the **Weak constraints** list and **Ideal** from the **Constraint type** list. These are the default settings and correspond to the most common type of boundary constraint for virtually all boundary conditions of the Dirichlet type. In the previous example, when using the pointwise ideal boundary constraint together with the integrated coupling variables, the coupling is bidirectional.

### *Weak Ideal Constraints*

---

The weak ideal boundary constraint gives the same result as if the weak constraints are turned off, except that using weak ideal constraints adds an additional dependent variable (known as a Lagrange multiplier), which you can use for extremely accurate postprocessing.

If you select **Ideal** from the **Constraint type** list in the previous example, you get the same solution. If you then open the **Solver Manager** and click the **Solve For** tab, the list of variables to solve for contains an additional dependent variable. The name of this Lagrange multiplier variable is `1m1`. The Lagrange multiplier is available for postprocessing via boundary integration on boundaries and provides an accurate value for the integrated flux.

### *Weak Non-Ideal Constraints*

---

The weak non-ideal boundary constraints gives a different solution. In this case the coupling is unidirectional, and it provides information transfer from the right boundary to the left boundary but not the other way around. The left boundary and the solution in the vicinity of the left boundary do not affect the solution at the right boundary. This type of unidirectional constraint can be explained intuitively as a

controller that senses the average temperature at the right boundary and forces the temperature at the left boundary to take this value. The left boundary has, however, no way of telling the right boundary of its temperature.

In general, use non-ideal boundary constraints for automatic control design and similar applications, where values from one part of the solution are “forced upon” another part of the solution. On the other hand, use ideal boundary constraints for problems where the model is “left by itself,” and the laws of nature balance the solution to an energy minimum (or, for problems with no well-defined energy, a saddle point).

Whenever using coupling variables in constraint expressions as in this example, make sure you do not use a different constraint at an adjacent boundary, because this would most likely create an ill-posed problem with no well-defined solution.

# Theoretical Background

The mathematical weak form gives you direct access to the terms of the weak equation and provides maximum freedom in defining finite element problems. This section provides a theoretical background to the weak form in COMSOL Multiphysics.

## *Deriving the Weak Form*

---

Before describing what the weak form is, this discussion first looks at the conversion of a simple problem in general form to the weak form.

Consider a stationary PDE problem for a single dependent variable,  $u$ , in two space dimensions:

$$\nabla \cdot \Gamma = F \quad \text{in } \Omega$$

Now let  $v$  be an arbitrary function on  $\Omega$ , and call it the *test function* ( $v$  should of course belong to a suitably chosen well-behaved class of functions,  $V$ ). Multiplying the PDE with this function and integrating leads to

$$\int_{\Omega} v \nabla \cdot \Gamma dA = \int_{\Omega} v F dA$$

where  $dA$  is the area element. Now use Green's formula (Gauss' formula) to integrate by parts:

$$\int_{\partial\Omega} v \Gamma \cdot \mathbf{n} ds - \int_{\Omega} \nabla v \cdot \Gamma dA = \int_{\Omega} v F dA$$

where  $ds$  is the length element. Then use the Neumann boundary condition

$$-\mathbf{n} \cdot \Gamma = G + \frac{\partial R}{\partial u} \mu$$

to arrive at the following equation:

$$0 = \int_{\Omega} (\nabla v \cdot \Gamma + v F) dA + \int_{\partial\Omega} v \left( G + \frac{\partial R}{\partial u} \mu \right) ds$$

Together with the Dirichlet condition, this is a weak reformulation of the original PDE problem. Note that the requirement is that the above weak equation should hold for *all* test functions  $v$ . You can reverse the steps of the derivation to show that if the functions  $u$  and  $\mu$  satisfy the weak formulation, then they also satisfy the original formulation. However, this holds true only if the solutions and coefficients are sufficiently smooth. For instance, in the case of discontinuities in material properties, you can have a solution of the weak formulation while the strong formulation has no meaning. The names “weak” and “strong” stem from this distinction: the weak formulation is a weaker condition on the solution than the strong formulation. A benefit of the weak formulation is that it requires less regularity of  $\Gamma$ . This is important in the finite element method.

For the most part, the weak reformulation of a system of equations in general form is

$$\left\{ \begin{array}{l} 0 = \int_{\Omega} (\nabla v_l \cdot \Gamma_l + v_l F_l) dA + \int_{\partial\Omega} v_l \left( G_l + \frac{\partial R_m}{\partial u_l} \mu_m \right) ds \\ 0 = R_m \quad \text{on } \partial\Omega \end{array} \right.$$

where you use the summation convention. If the boundary conditions are given on interior boundaries,  $\partial\Omega$  can be enlarged to include these. You now have several test functions  $v_1, v_2, \dots, v_N$ , one for each of the original PDEs. The weak equation is required to hold for all test functions (in the suitably chosen function space  $V$ ). Of course, in three dimensions, for  $dA$  and  $ds$  you must substitute the volume and surface area elements, respectively. In one dimension, for  $dA$  substitute the length element, and interpret the boundary integral as a sum over the boundary points.

### *Weak Form Application Modes*

---

In COMSOL Multiphysics you can add weak form PDEs on all domain levels. From the **PDE Modes** folder in the Model Navigator, select from:

- **Weak Form, Subdomain**
- **Weak Form, Boundary**
- **Weak Form, Edge** (3D only)
- **Weak Form, Point**

The Weak Form, Boundary application modes are of great use when you must define a PDE on a boundary and couple this equation to an equation in a subdomain. A

typical example exists in adsorption processes when you are coupling a material balance of the adsorbed species on the boundary to the material balance of a species defined in the domain (see “Transport and Adsorption” on page 32 of the *COMSOL Multiphysics Model Library*).

In addition, you can add weak form contributions to other application modes on the **Weak** page in the **Subdomain Settings**, **Boundary Settings**, **Edge Settings**, and **Point Settings** dialog boxes.

For an example of how to use a weak form contribution to implement a point source, see “Implementing a Point Source” on page 285.

### VARIATIONAL PRINCIPLES

This section provides some background on the weak formulation. You can eliminate the Lagrange multipliers from the weak equation in the previous section by introducing the following boundary conditions on the test functions:

$$0 = \frac{\partial R_m}{\partial u_l} v_l \quad \text{on } \partial\Omega$$

This means that the functions  $u_l + v_l$  satisfy the linearized version of the Dirichlet boundary conditions. Then the weak formulation becomes the following: find functions  $u_l$  such that

$$\begin{cases} 0 = \int_{\Omega} (\nabla v_l \cdot \Gamma_l + v_l F_l) dA + \int_{\partial\Omega} v_l G_l ds \\ 0 = R_m \quad \text{on } \partial\Omega \end{cases}$$

holds for all test functions  $v_l$  that satisfy the above boundary condition. Such a formulation arises when you have a *variational principle*; for instance, to find the functions  $u_l$  that minimizes the energy of some physical system under the constraints  $0 = R_m$ . If the energy is given as an integral of some expression involving the  $u_l$ , then the stationarity condition on the solution is precisely the weak formulation above. Because variational principles are more fundamental than the corresponding PDE, the weak form is often more natural than the strong forms. The connection to variational principles also shows why the term involving the Lagrange multipliers takes the form it does (that is, how ideal constraints come about).

## THE WEAK FORM PDE STRUCTURE

Now consider what a problem on the weak form can look like. It can be more general than the weak equations derived from the general form above because it also contains integrals over edges (in 3D) or points (in 2D and 3D). Similarly, you can also have constraints on subdomains, edges, and points. Further, expressions occurring in the integrands can be more or less arbitrary functions of the variables  $u_l$ , the test functions  $v_l$ , and their derivatives. One important restriction, though, is that the test functions enter linearly. A problem in weak form has the following structure (in 2D):

$$\begin{aligned}
 0 = & \int_{\Omega} (W^{(2)} - W^{(2t)}) dA + \int_B (W^{(1)} - W^{(1t)}) ds + \sum_P (W^{(0)} - W^{(0t)}) + \\
 & + \int_{\Omega} v_l \frac{\partial R_m^{(2)}}{\partial u_l} \mu_m^{(2)} dA + \int_B v_l \frac{\partial R_m^{(1)}}{\partial u_l} \mu_m^{(1)} ds + \sum_P v_l \frac{\partial R_m^{(0)}}{\partial u_l} \mu_m^{(0)} \\
 & 0 = R^{(2)} \quad \text{on } \Omega \\
 & 0 = R^{(1)} \quad \text{on } B \\
 & 0 = R^{(0)} \quad \text{on } P
 \end{aligned}$$

The first equation is the weak equation, and the others are the constraints. Here  $\Omega$  is the subdomain, B is the boundary (including  $\partial\Omega$  and interior boundaries), and P represents the vertices (points) defined in the geometry. The integrands  $W^{(n)}$ , which correspond to the sum of all **weak** field entries for dimension  $n$ , are scalar expressions involving the dependent variables  $u_1, u_2, \dots, u_N$  as well as the test functions  $v_1, v_2, \dots, v_N$  and their derivatives. You must enter the test functions and their derivatives linearly. For instance, you can have

$$W^{(2)} = v_1 \frac{\partial u_2}{\partial x} + \frac{\partial v_2}{\partial x} \sin u_1 + \frac{\partial^2 v_1}{\partial y^2}$$

The integrands  $W^{(nt)}$  which correspond to the **dweak** fields have exactly the same form as the  $W^{(n)}$ , but enter with opposite sign.

The quantities  $R^{(i)}$  occurring in the constraints are vector-valued functions of the dependent variables. Sometimes they also depend on the derivatives of the dependent variables. In such cases there are additional terms accounting for this dependence in the integrals with Lagrange multipliers. Note that you now also can have Lagrange multipliers on subdomains ( $\mu^{(2)}$ ) and on vertices ( $\mu^{(0)}$ ).

In 3D, contributions to the weak equation and the constraints can also come from edges (curves).

### CONVERSION TO WEAK FORM

It is possible to convert equations in coefficient form or general form to weak form by setting the equation system form to weak using the **Equation system form** list in the **Model Settings** dialog box, which opens from the **Physics** menu. To view the result of the conversion, go to the **Physics** menu, point to **Equation System** and then click on a domain type settings command. COMSOL Multiphysics also carries out this conversion conceptually in the finite element method. The following equations describe the obtained weak form problem as well as problems given directly in weak form. For simplicity this discussion considers a stationary problem in 2D.

$$\begin{aligned}
 0 &= \int_{\Omega} (W^{(2)} - W^{(2t)}) dA + \int_B (W^{(1)} - W^{(1t)}) ds + \sum_P (W^{(0)} - W^{(0t)}) \\
 &\quad - \int_{\Omega} v \cdot h^{(2)T} \mu^{(2)} dA - \int_B v \cdot h^{(1)T} \mu^{(1)} ds - \sum_P v \cdot h^{(0)T} \mu^{(0)} \\
 0 &= R^{(2)} \quad \text{on } \Omega \\
 0 &= R^{(1)} \quad \text{on } B \\
 0 &= R^{(0)} \quad \text{on } P
 \end{aligned}$$

Here  $v$  is a vector whose components are the test functions  $v_l$ , while  $\mu^{(d)}$  are column vectors of Lagrange multipliers  $\mu_m^{(d)}$ . The elements of matrices  $h^{(d)}$  are

$$h_{ml}^{(d)} = \frac{\partial R_m^{(d)}}{\partial u_l}$$

for a problem given directly in weak form. For a problem originally given in coefficient form or general form, the vectors  $R^{(2)}$  and  $R^{(0)}$  as well as the Lagrange multipliers  $\mu^{(2)}$  and  $\mu^{(0)}$  are empty. In addition,  $W^{(0)} = 0$ . For the general form you have

$$h_{lm}^{(1)} = \frac{\partial R_m}{\partial u_l}$$

For the coefficient form, you have  $h^{(1)} = h$ .

For a problem given directly in weak form, the constraints can also depend on variables other than the  $u_l$  (for instance the space derivatives of  $u_l$ ). In such cases, the quantities  $v \cdot h^{(d)T}$  are augmented with additional terms that account for this dependence.



Conversion from general form to weak form is performed according to

$$W^{(n)} = \Gamma_{lj} \frac{\partial v_l}{\partial x_j} + F_l v_l + \sum_l W_l^{0(n)}$$

$$W^{(nt)} = d_{alk} \frac{\partial u_k}{\partial t} v_l + \sum_l W_l^{0(nt)}$$

$$W^{(n-1)} = G_l v_l + \sum_l W_l^{0(n-1)}$$

$$R_m^{(n)} = R_m$$

where there is an implicit summation over repeated indices in each product. The superscript  $(n)$  is the space dimension, while the superscript 0 marks explicit weak contributions added to the general form equation. Note that explicit time derivative terms for historical reasons are collected in the  $W^{(nt)}$ . There is nothing prevent you from entering time derivatives in  $W^{(n)}$ , though.

If you are working with COMSOL Script or MATLAB, see the entry `f1form` in the *COMSOL Multiphysics Reference Guide* for more information about the conversion to weak form.

### *Entering a PDE in Coefficient Form Using the Weak Form*

---

Derive the weak equation for a coefficient form problem:

$$0 = \int_{\Omega} (\nabla v \cdot (-c \nabla u - \alpha u + \gamma) + v(f - \beta \cdot \nabla u - \alpha u)) dA + \int_{\partial\Omega} v(-qu + g - h^T \mu) ds$$

Suppose  $c$  is a scalar. Using the weak form meta variables for a 2D problem, follow these steps:

- 1 In the **weak** edit field of  $\Omega$  in the **Subdomain Settings** dialog box, type
 

```
ux_test*(-c*ux-ax*u+gax)+uy_test*(-c*uy-aly*u+gay)+...
u_test*(f-bex*ux-bey*uy-a*u)
```
- 2 In the **weak** field of  $\partial\Omega$  in the **Boundary Settings** dialog box, type
 

```
u_test*(-q*u+g)
```

The variables  $a_{1x}$ ,  $a_{1y}$ ,  $b_{ex}$ ,  $b_{ey}$ ,  $g_{ax}$ , and  $g_{ay}$  are the components of  $\alpha$ ,  $\beta$ , and  $\gamma$ , respectively. You must replace the coefficients in this weak form expression (for example,  $c$ ,  $a_{1x}$  and  $q$ ) with appropriate expressions.

COMSOL Multiphysics automatically adds the Lagrange multiplier  $\mu$ , while the constraint matrix  $h$  is specified indirectly by the constraints.

# Multiphysics Modeling

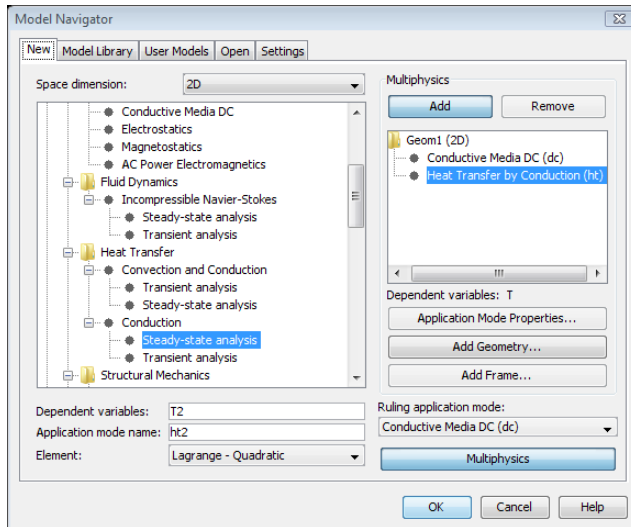
This chapter covers the use of COMSOL Multiphysics for multiphysics modeling and coupled-field analyses. It first describes the various ways of building multiphysics models. Then a step-by-step example shows how to model Joule heating (also called resistive heating) taking into account thermal-electric couplings.

# Creating Multiphysics Models

## *Multiphysics Modeling Approaches*

The ability to create multiphysics models—those with more than one type of physics or equation such as coupled-field problems—is one of COMSOL Multiphysics’ most powerful capabilities. In such a model, the software can solve all the equations, taken from various areas of physics, as one fully coupled system.

Within COMSOL Multiphysics you can choose from several ways to approach multiphysics modeling and coupled-field analysis. They all use the **Model Navigator** to specify the application modes and their properties.



*Figure 14-1: The Model Navigator for a multiphysics model combining the Conductive Media DC and Heat Transfer by Conduction application modes.*

## *Using Predefined Multiphysics Couplings*

COMSOL Multiphysics and the add-on modules provide a number of predefined multiphysics couplings. They appear in the Model Navigator’s list of application modes, in folders with the names in the Category/Folder column in Table 14-1. For each predefined multiphysics coupling, there is an entry in the Model Navigator in all products that support it. The predefined multiphysics couplings consist of two or more

application modes that COMSOL Multiphysics adds to a model. In addition, default settings provide the typical field couplings for the multiphysics application. The available predefined multiphysics couplings vary depending on the installed modules (see the following table for details).

TABLE 14-1: PREDEFINED MULTIPHYSICS COUPLINGS

CATEGORY/ FOLDER	NAME	SPACE DIMENSION	REQUIRED MODULES	OPTIONAL MODULES
Electro-Thermal Interaction	Joule Heating	2D, 2D Axi, 3D	None	AC/DC, Heat Transfer, MEMS
Electro-Thermal Interaction	Perpendicular Induction Heating	2D	AC/DC	Heat Transfer
Electro-Thermal Interaction	In-Plane Induction Heating	2D	AC/DC	Heat Transfer
Electro-Thermal Interaction	In-Plane Microwave Heating	2D	RF	Heat Transfer
Electro-Thermal Interaction	Meridional Induction Heating	2D Axi	AC/DC	Heat Transfer
Electro-Thermal Interaction	Azimuthal Induction Heating	2D Axi	AC/DC	Heat Transfer
Electro-Thermal Interaction	Meridional Microwave Heating	2D Axi	RF	Heat Transfer
Electro-Thermal Interaction	Induction Heating	3D	AC/DC	Heat Transfer
Electro-Thermal Interaction	Microwave Heating	3D	RF	Heat Transfer
Fluid-Chemical Reactions	Reacting Flow	2D, 2D Axi, 3D	Chemical Engineering	
Fluid-Structure Interaction	Plane Strain with Fluid Interaction	2D	Structural Mechanics or MEMS	Chemical Engineering
Fluid-Structure Interaction	Axial Symmetry, Stress-Strain with Fluid Interaction	2D Axi	Structural Mechanics or MEMS	Chemical Engineering
Fluid-Structure Interaction	Solid, Stress-Strain with Fluid Interaction	3D	Structural Mechanics or MEMS	Chemical Engineering

TABLE 14-1: PREDEFINED MULTIPHYSICS COUPLINGS

CATEGORY/ FOLDER	NAME	SPACE DIMENSION	REQUIRED MODULES	OPTIONAL MODULES
Fluid-Thermal Interaction	Fluid-Thermal Incompressible Flow	2D, 2D Axi, 3D	None	
Fluid-Thermal Interaction/Flow with Variable Density	Non-Isothermal Flow (laminar)	2D, 2D Axi, 3D	Chemical Engineering or Heat Transfer	
Fluid-Thermal Interaction/Flow with Variable Density	Turbulent Non-Isothermal Flow	2D, 2D Axi, 3D	Chemical Engineering or Heat Transfer	
Thermal-Structural Interaction	Plane Stress with Thermal Expansion	2D	Structural Mechanics or MEMS	Heat Transfer
Thermal-Structural Interaction	Plane Strain with Thermal Expansion	2D	Structural Mechanics or MEMS	Heat Transfer
Thermal-Structural Interaction	Axial Symmetry, Stress-Strain with Thermal Expansion	2D Axi	Structural Mechanics or MEMS	Heat Transfer
Thermal-Structural Interaction	Solid, Stress-Strain with Thermal Expansion	3D	Structural Mechanics or MEMS	Heat Transfer
Thermal-Structural Interaction	Shell with Thermal Expansion	3D	Structural Mechanics or MEMS, Heat Transfer	
Thermal-Electric-Structural Interaction	Plane Stress with Thermal-Electric Interaction	2D	Structural Mechanics or MEMS	Heat Transfer, AC/DC
Thermal-Electric-Structural Interaction	Plane Strain with Thermal-Electric Interaction	2D	Structural Mechanics or MEMS	Heat Transfer, AC/DC

TABLE 14-1: PREDEFINED MULTIPHYSICS COUPLINGS

CATEGORY/ FOLDER	NAME	SPACE DIMENSION	REQUIRED MODULES	OPTIONAL MODULES
Thermal-Electric-Structural Interaction	Axial Symmetry, Stress Strain with Thermal-Electric Interaction	2D Axi	Structural Mechanics or MEMS	Heat Transfer, AC/DC
Thermal-Electric-Structural Interaction	Solid, Stress Strain with Thermal-Electric Interaction	3D	Structural Mechanics or MEMS	Heat Transfer, AC/DC
Acoustic-Structure Interaction	Plane Strain with Acoustic Interaction	2D	Acoustics	Structural Mechanics, MEMS
Acoustic-Structure Interaction	Axial Symmetry, Stress-Strain with Acoustic Interaction	2D Axi	Acoustics	Structural Mechanics, MEMS
Acoustic-Structure Interaction	Solid, Stress-Strain with Acoustic Interaction	3D	Acoustics	Structural Mechanics, MEMS
Structural Mechanics	Plane Strain with Film Damping	2D	MEMS	
Structural Mechanics	Axial Symmetry, Stress-Strain with Film Damping	2D Axi	MEMS	
Structural Mechanics	Solid, Stress-Strain with Film Damping	3D	MEMS	
Microfluidics	Flow with Species Transport	2D, 2D Axi, 3D	MEMS	
Microfluidics	Electroosmotic Flow	2D, 2D Axi, 3D	MEMS	
Rotating Machinery	Rotating Perpendicular Currents	2D	AC/DC	
Rotating Machinery	Rotating Navier-Stokes	2D	Chemical Engineering	

TABLE 14-1: PREDEFINED MULTIPHYSICS COUPLINGS

CATEGORY/ FOLDER	NAME	SPACE DIMENSION	REQUIRED MODULES	OPTIONAL MODULES
Poroelasticity	Plane Strain, Pressure/Pressure head/Hydraulic head	2D	Earth Science, Structural Mechanics/ MEMS	
Poroelasticity	Stress-Strain, Pressure/Pressure head/Hydraulic head	2D Axi	Earth Science, Structural Mechanics/ MEMS	
Poroelasticity	Stress-Strain, Pressure/Pressure head/Hydraulic head	3D	Earth Science, Structural Mechanics/ MEMS	

The predefined multiphysics coupling makes use of an application mode in the optional modules if your license includes them; otherwise the coupling uses the corresponding but more limited application mode in COMSOL Multiphysics or the required module. The couplings are identical in both cases, but the application mode from the optional module typically offers additional functionality in other areas. These predefined multiphysics couplings are quick entry points for common multiphysics applications. It is possible to create the same couplings using any of the other methods for multiphysics modeling, and you can continue to add, modify, and remove application modes in a model that you start using one of the predefined multiphysics couplings. If you want to add additional application modes directly in the **Model Navigator**, click the **Multiphysics** button and then the **Add** button. You can then see the application modes from the predefined multiphysics coupling in the **Multiphysics** area and select new application modes that you add to the model using the **Add** button.

In the **Model Navigator**, the **Dependent variables** list contains the dependent variables, and the **Application mode name** edit field contains the default application mode names for the application modes that the predefined multiphysics coupling includes. The **Element** list in the **Model Navigator** provides a selection of suitable element types for all the dependent variables. You can also select element types individually using the **Element** tab in the **Subdomain Settings** dialog box (and, in some cases, the **Boundary Settings** dialog box).

### ELECTRO-THERMAL INTERACTION—JOULE HEATING

The Joule Heating predefined multiphysics coupling combines a Conductive Media DC application mode from COMSOL Multiphysics or the AC/DC Module with the



Heat Transfer by Conduction application mode from COMSOL Multiphysics or the General Heat Transfer application mode from the Heat Transfer Module. The interaction is fully coupled in both directions:

- In the heat transfer application mode, resistive heating appears as a heat source (the resistive heating variable  $Q_{dc}$  (or  $Q_{emdc}$ ) appears in the  $Q$  edit field).
- In the Conductive Media DC application mode, the definition of the conductivity  $\sigma$  uses the thermal heating model  $1/(\rho_0(1+\alpha(T-T_0)))$ , where  $T$  is the dependent variable for temperature from the heat transfer application mode. The values for  $\rho_0$  (resistivity at reference temperature),  $\alpha$  (temperature coefficient), and  $T_0$  (reference temperature) are default values that you can change to match your modeling situation.

You can select a steady-state or a transient analysis.

For an example of a model using the Joule Heating predefined multiphysics coupling, see “Example—Resistive Heating” on page 395.

#### **ELECTRO-THERMAL INTERACTION—INDUCTION HEATING**

The Induction Heating predefined multiphysics coupling combines an application mode for induction currents from the AC/DC Module with the Heat Transfer by Conduction application mode from COMSOL Multiphysics or the General Heat Transfer application mode from the Heat Transfer Module. The predefined interaction adds the resistive heating from the induction currents as a heat source in the application mode for the heat transfer.

For more information, see the *AC/DC Module User's Guide*.

#### **ELECTRO-THERMAL INTERACTION—MICROWAVE HEATING**

The Microwave Heating predefined multiphysics coupling combines an application mode for electromagnetic waves from the RF Module with the Heat Transfer by Conduction application mode from COMSOL Multiphysics or the General Heat Transfer application mode from the Heat Transfer Module. The predefined interaction adds the resistive heating from the electromagnetic waves as a heat source in the application mode for the heat transfer.

For more information, see the *RF Module User's Guide*.

#### **FLUID-CHEMICAL REACTIONS INTERACTION—REACTING FLOW**

The Reacting Flow predefined multiphysics coupling requires the Chemical Engineering Module. Reacting Flow combines a Convection and Diffusion mass

balance application mode with an Incompressible Navier-Stokes application mode, both from the Chemical Engineering Module. The interaction is a one-way coupling using the velocity described by the Navier-Stokes equations as part of the convective mass transfer. The components of the velocity vector  $\mathbf{u}$  from the Incompressible Navier-Stokes application mode appear in the  $\mathbf{u}$ ,  $\mathbf{v}$ , and  $\mathbf{w}$  edit fields in the **Subdomain Settings** dialog box for the Convection and Diffusion application mode.

By default, COMSOL Multiphysics solves for the velocity field, pressure, and species concentrations simultaneously. For large problems, you can take advantage of the one-way coupling and solve the problem sequentially (unless there are fluid-flow properties that depend on the species concentrations): First solve for velocities and pressures, and then solve the mass balance equation using the computed velocities from the Navier-Stokes equation. See “Solving for a Subset of the Dependent Variables” on page 389 on how to select which variables to solve for.

#### **FLUID-STRUCTURE INTERACTION (FSI)**

The Fluid-Structure Interaction (FSI) predefined multiphysics coupling is available in the MEMS Module and the Structural Mechanics Module. FSI combines fluid flow with structural mechanics, using a Moving Mesh (ALE) application mode to capture the fluid movement. Predefined groups of boundary settings help to define the fluid loads on the solid domain and the structural velocity’s effect on the fluid. For more information and example models, see the Structural Mechanics Module and MEMS Module documentation.

#### **FLUID-THERMAL INTERACTION—FLUID-THERMAL INCOMPRESSIBLE FLOW**

The Fluid-Thermal Incompressible Flow predefined multiphysics coupling in COMSOL Multiphysics combines an Incompressible Navier-Stokes application mode with the Convection and Conduction application mode. The interaction is a one-way coupling using the velocity described by the Navier-Stokes equations as part of the convective heat transfer. The components of the velocity vector  $\mathbf{u}$  from the Incompressible Navier-Stokes application mode appear in the  $\mathbf{u}$ ,  $\mathbf{v}$ , and  $\mathbf{w}$  edit fields in the **Subdomain Settings** dialog box for the Convection and Conduction application mode.

By default, COMSOL Multiphysics solves for the velocity field, pressure, and species concentrations simultaneously. For large problems, you can take advantage of the one-way coupling and solve the problem sequentially (unless there are fluid-flow properties that depend on the temperature): first solve for velocities and pressures and then solve the heat transfer equation using the computed velocities from the

Navier-Stokes equation. See “Solving for a Subset of the Dependent Variables” on page 389 on how to select which variables to solve for.

#### **FLOW WITH VARIABLE DENSITY—NON-ISOTHERMAL FLOW**

The Chemical Engineering Module and the Heat Transfer Module include the Non-Isothermal Flow and Turbulent Non-Isothermal Flow predefined multiphysics couplings (in the Chemical Engineering Module you select both laminar and turbulent non-isothermal flow in the **Flow with Variable Density>Non-Isothermal Flow** folder; in the Heat Transfer Module, you select a laminar or turbulent non-isothermal flow in the **Fluid-Thermal Interaction** folder), which are similar but model non-isothermal laminar and turbulent fluids. For more information about these predefined couplings, see the Chemical Engineering Module and Heat Transfer Module documentation.

#### **THERMAL-STRUCTURAL INTERACTION—THERMAL EXPANSION**

The predefined multiphysics couplings for structural mechanics with thermal expansion requires the Structural Mechanics Module or the MEMS Module. They combine the Heat Transfer by Conduction application mode from COMSOL Multiphysics or the General Heat Transfer application mode from the Heat Transfer Module with the following application modes from the Structural Mechanics Module or the MEMS Module:

- Plane Stress in 2D
- Plane Strain in 2D
- Axial Symmetry, Stress-Strain in 2D axisymmetry
- Solid, Stress-Strain in 3D

The interaction is a one-way coupling using the temperature described by a heat transfer application mode to define the strain temperature. On the **Load** page in the **Subdomain Settings** dialog box in the structural mechanics application mode, the **Include thermal expansion** check box is selected, and the dependent variable for temperature from the heat transfer application mode appears in the **Temp** edit field for the strain temperature. The **Tempref** edit field contains a default value (0) for the strain reference temperature. If necessary, adjust the strain reference temperature to a suitable value for your model.

You can select a steady-state, transient, or quasi-static analysis.

By default, COMSOL Multiphysics solves for the temperature and displacements simultaneously. For large problems, you can take advantage of the one-way coupling and solve the problem sequentially (unless there are thermal properties that depend on

the displacements): first solve for temperature and then perform the stress-strain analysis using the computed temperature field from the heat transfer equation. See “Solving for a Subset of the Dependent Variables” on page 389 on how to select which variables to solve for.

### **THERMAL-ELECTRIC-STRUCTURAL INTERACTION**

The predefined multiphysics couplings for structural mechanics with thermal-electric interaction requires the Structural Mechanics Module or the MEMS Module. They combine the Heat Transfer by Conduction application mode from COMSOL Multiphysics or the General Heat Transfer application mode from the Heat Transfer Module plus the Conductive Media DC application mode with the following application modes from the Structural Mechanics Module or the MEMS Module:

- Plane Stress in 2D
- Plane Strain in 2D
- Axial Symmetry, Stress-Strain in 2D axisymmetry
- Solid, Stress-Strain in 3D

The interactions include Joule heating, which is fully coupled in both directions (see “Electro-Thermal Interaction—Joule Heating” on page 380), and thermal expansion, which is a one-way coupling (see “Thermal-Structural Interaction—Thermal Expansion” on page 383).

- In the heat transfer application mode, resistive heating appears as a heat source (the resistive heating variable  $Q_{dc}$  (or  $Q_{emdc}$ ) appears in the **Q** edit field).
- In the Conductive Media DC application mode, the definition of the conductivity  $\sigma$  uses the thermal heating model  $1/(\rho_0(1+\alpha(T-T_0)))$ , where  $T$  is the dependent variable for temperature from the heat transfer application mode. The values for  $\rho_0$  (resistivity at reference temperature),  $\alpha$  (temperature coefficient), and  $T_0$  (reference temperature) are default values that you can change to match your modeling situation.

You can select a steady-state or transient analysis (where the conductive media and structural mechanics equations are stationary).

By default, COMSOL Multiphysics solves for the temperature, electric potential, and displacements simultaneously.

### **ACOUSTIC-STRUCTURE INTERACTION**

The predefined multiphysics coupling for acoustic-structural interaction is available in the Acoustics Module and in the Structural Mechanics Module and the MEMS Module, if the license also includes the Acoustics Module. This multiphysics coupling connects the acoustics pressure in a fluid domain with the structural deformation in a solid domain. Predefined groups of boundary settings help to define the fluid loads on the solid domain and the structural acceleration's effect on the fluid.

For more information, see the Acoustics Module documentation.

### **STRUCTURAL MECHANICS INCLUDING FILM DAMPING**

The MEMS Module includes the Plane Strain with Film Damping and Solid, Stress-Strain with Film Damping predefined multiphysics couplings. The couple the structural mechanics application mode with the Film Damping application mode, for transient and frequency-response analysis of microsystems that include film damping. The Film Damping application mode models film damping on boundaries, and the predefined multiphysics coupling sets the boundary deformation to the displacement variables from the structural mechanics application mode.

For more information, see the MEMS Module documentation.

### **MICROFLUIDICS—FLOW WITH SPECIES TRANSPORT**

This suite of predefined multiphysics couplings combines the application modes for microfluidics (General Laminar Flow, Incompressible Navier-Stokes, Non-Isothermal Flow, Stokes Flow, and Non-Isothermal Stokes Flow) with the Convection and Diffusion application mode. The coupling is similar to the one in the Reacting Flow predefined multiphysics coupling (see “Fluid-Chemical Reactions Interaction—Reacting Flow” on page 381), combining fluid flow with species (mass) transport.

For more information, see the MEMS Module documentation.

### **MICROFLUIDICS—ELECTROOSMOTIC FLOW**

This suite of predefined multiphysics couplings for electroosmotic flow modeling combines the application modes for microfluidics (General Laminar Flow, Incompressible Navier-Stokes, Non-Isothermal Flow, Stokes Flow, and Non-Isothermal Stokes Flow) with the Conductive Media DC application mode.

Using the Electroosmotic Flow predefined multiphysics coupling, both application modes include boundary groups for defining boundaries as electrodes, ground, electroosmosis, inlets, and outlets.

For more information, see the MEMS Module documentation.

#### **ROTATING MACHINERY—ROTATING PERPENDICULAR CURRENTS**

For modeling of rotating machinery such as motors and generators, the Rotating Perpendicular Currents predefined multiphysics coupling combines the Perpendicular Induction Currents application mode with a Moving Mesh (ALE) application mode. The Moving Mesh (ALE) application mode, which captures the rotation, contains predefined groups with settings for fixed parts and parts that rotate clockwise or counterclockwise.

For more information, see the AC/DC Module documentation.

#### **ROTATING MACHINERY—ROTATING NAVIER-STOKES**

For modeling of rotating machinery with fluids (such as stirrers), the Rotating Navier-Stokes predefined multiphysics coupling combines the Incompressible Navier-Stokes application mode with a Moving Mesh (ALE) application mode. The Moving Mesh (ALE) application mode, which captures the rotation, contains predefined groups with settings for fixed parts and parts that rotate clockwise or counterclockwise.

For more information, see the Chemical Engineering Module documentation.

#### **POROELASTICITY**

This predefined multiphysics coupling is available in the Earth Science Module and combines the Darcy's Law application mode from that module with a structural mechanics application mode from the Structural Mechanics Module or the MEMS Module (one of those modules are required in addition to the Earth Science Module). The Poroelasticity predefined multiphysics coupling is useful for modeling poroelastic phenomena such as diffusion in elastic solids or elasticity of fluid-infiltrated porous solids. For more information, see the Earth Science Module documentation.

### *Adding Physics Sequentially*

---

With this approach you can verify that each type of physics or PDE gives the expected results before adding more complexity to the model by adding another physics or coupling fields.

Use these steps to add one application mode at the time:

- 1 Select an application mode.
- 2 Draw the geometry.

- 3 Define the physics settings.
- 4 Solve and visualize the results.

Then add another application mode:

- 5 Go to the **Multiphysics** menu and choose **Model Navigator**.
- 6 Select a new application mode.
- 7 In the **Multiphysics** area, click the **Add** button.
- 8 Click **OK**.
- 9 Add and modify settings in the **Physics** menu, including couplings between the physics.

### *Building a Coupled Multiphysics Model Directly*

---

Another approach is to include multiple application modes when you start creating the model in the **Model Navigator**:

- 1 Select an application mode for the model.
- 2 Click the **Multiphysics** button.
- 3 In the **Multiphysics** area, click the **Add** button.
- 4 Continue selecting application modes and adding them to the model by clicking **Add**.
- 5 Click **OK**.

### *Removing Application Modes*

---

To remove an application mode from a model:

- 1 Choose **Model Navigator** from the **Multiphysics** menu.
- 2 Select the application mode from the list under **Multiphysics**. You can also select a geometry in the list to remove the entire geometry and all application modes that it contains.
- 3 Click **Remove**.
- 4 Click **OK**.

## *Multiphysics Model Properties*

---

When you add or remove application modes, COMSOL Multiphysics updates some model properties to:

- Include all dependent variables in the added application modes in the list of variables to solve for.
- Update all application-dependent dialog boxes.

### **ACTIVE APPLICATION MODE**

The application mode selected under **Multiphysics** becomes the active application mode when you begin modeling.

To shift between the different application modes during modeling, select one from the **Multiphysics** menu to make it active.

### **RULING APPLICATION MODE**

The *ruling application mode* determines the global settings for the multiphysics model from its application mode properties. For example, if the analysis type is different for two application modes in the same model, the software selects the solver based on the analysis type from the ruling application mode. You select the ruling application mode for a multiphysics model in the Model Navigator. In the **Solver Parameters** dialog box, you can see (and change) the analysis types for all application modes in the **Analysis types** area. In this area, the ruling application mode appears on top using a bold font.

## *Managing the Solver and Solution Components*

---

The Solver Manager dialog box provides several options for managing the solution process for multiphysics analyses.

### **RESTARTING THE SOLVER AND USING INITIAL VALUES**

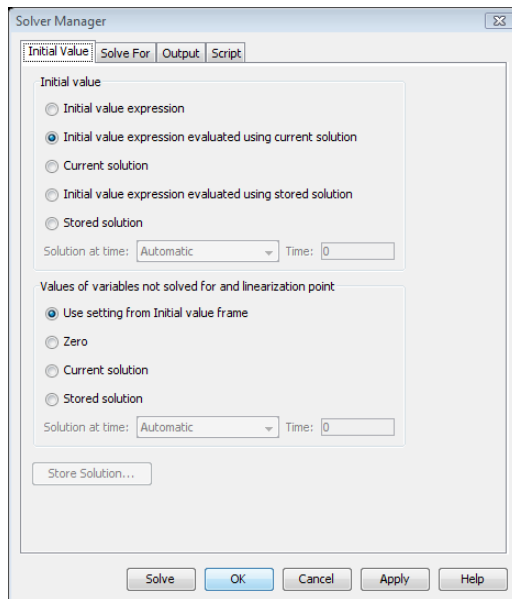
When adding equations and physics to an existing model, you can restart the model using the current solution as an initial value. Doing so can accelerate the convergence for a coupled multiphysics model. To restart the solver, click the **Restart** button on the Main toolbar.

To specify initial values:

- 1 Go to the **Solve** menu and choose **Solver Manager**.
- 2 In the **Solver Manager** dialog box click the **Initial Value** tab.



- 3 Click the **Initial value expression using current solution** button (the default setting) to evaluate the expressions for the initial values as specified for each application mode using the current solution.
- 4 Click **OK**.



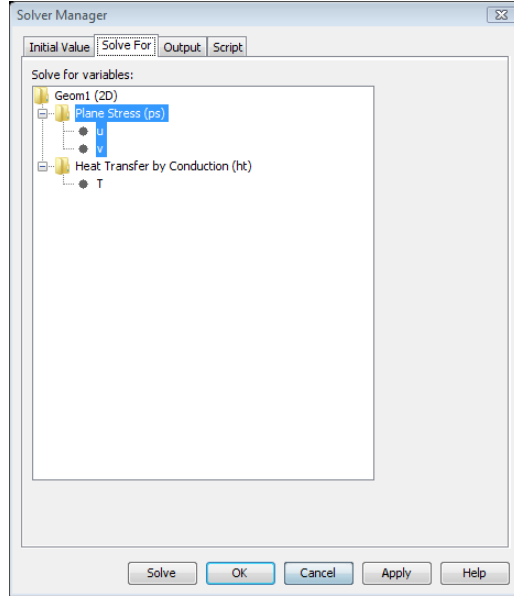
*Figure 14-2: The Solver Manager settings for evaluation the initial value expressions using the current solution.*

### **SOLVING FOR A SUBSET OF THE DEPENDENT VARIABLES**

The solver normally solves for all dependent variables. It is also possible to have it solve for a subset of the dependent variables. The initial conditions then define the value of the other variables. To specify which variables to solve for:

- 1 Go to the **Solve** menu and choose the **Solver Manager**.
- 2 In the **Solver Manager** dialog box click the **Solve For** tab.
- 3 Within the **Solve for variables** list the software selects all application modes by default. Clear the selections for those application modes whose dependent variables you do not want to solve for.
- 4 To select individual dependent variables for an application mode, click the plus sign to open the application mode folder and select the dependent variables to include.

5 Click **OK**.



*Figure 14-3: The Solve For page of the Solver Manager set up to solve for the displacements in the Plane Stress application model only.*

Use this subset approach to reduce the computational load when there is only a one-way coupling in the multiphysics model:

- 1 Solve for the variables that are not affected by the coupling.
- 2 Solve for the remaining variables, using the solution from the initial analysis by restarting the solver using the previous solution data.

For a fully coupled multiphysics model with bidirectional couplings you can also use this approach to reach convergence in an iterative manner. Solve for one type of physics at a time and then use that solution as the initial value when solving for the other type of physics. This method provides a “weakly coupled” way of solving a coupled multiphysics model. Using the standard COMSOL Multiphysics approach to solving fully coupled equations simultaneously normally provides better convergence. The stationary and time-dependent segregated solvers provide an efficient way of solving fully coupled and “one-way coupled” multiphysics models. Several predefined multiphysics couplings provide suitable default settings using the segregated solvers.

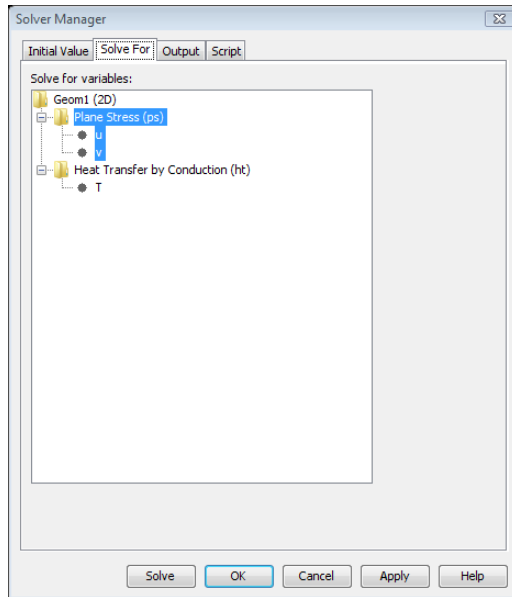
## USING VARIABLES DURING ANALYSIS ONLY

In a similar way you can choose to output the results for a subset of the dependent variables:

- 1 In the **Solver Manager** dialog box click the **Output** tab.
- 2 In the **Output for variables** list select from the application modes and their dependent variables in the same way as for the variables to solve for.
- 3 Click **OK**.

This method can be useful if you want to include variables for multiphysics couplings during the analysis but are not interested in using them for postprocessing.

If you include nonsolved variables for output they take on the specified initial value as their solution data in the output from the solver.



*Figure 14-4: The Output page of the Solver Manager. By default, COMSOL Multiphysics includes all dependent variables from all application modes in the results data.*

### *Deactivating Physics or Equations*

Sometimes subdomains have different types of physics associated with them, and an application mode might not be meaningful in a subdomain. For example, fluid flow can be present only in some subdomains of a model whereas heat transfer takes place

in the entire geometry. By default all application modes are active in all subdomains. To deactivate an application mode, select the subdomains where the application mode's physics do not take place. Then clear the **Active in this subdomain** check box in the **Subdomain Settings** dialog box and click **Apply**. Some interior boundaries then become exterior boundaries, and some of the original boundaries are absent for certain application modes.

### *Using the Model Navigator for Multiphysics Models*

---

To add and remove application modes and geometries for a multiphysics model, go to the **Multiphysics** menu and then open the **Model Navigator**. Use its **Multiphysics** area to make changes to the current model's multiphysics properties.

#### **ADDING GEOMETRIES TO A MODEL**

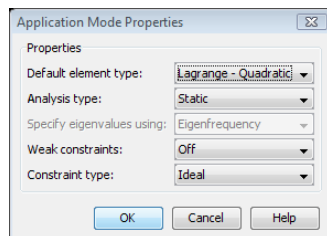
A model always has at least one geometry. To add another geometry to a model, click the **Add Geometry** button in the **Multiphysics** area of the **Model Navigator**. In such an *extended multiphysics* model the geometries can have different space dimensions. You can also choose the names of the independent variables (spatial coordinates).

#### **CHANGING APPLICATION MODE PROPERTIES**

Before adding an application mode, you can edit its name and dependent variables. COMSOL Multiphysics provides a unique name for each application mode in the model to identify the origin of the variables. You can change the name to any other name that is unique within the model.

You can also specify the element type, although the default is usually a good choice.

To set the analysis type and other application mode properties for the active application mode in a multiphysics model, click the **Application Mode Properties** button. These properties vary depending on the application mode.



*Figure 14-5: The Application Mode Properties dialog box. The contents vary depending on the active application mode.*

## *Specifying Settings for Multiphysics Models*

---

In multiphysics models with multiple application modes, you can work with the settings for more than one application mode at a time. Selecting a settings command from the **Physics** menu opens the dialog box for the current application mode. All application modes in the model are listed in the **Multiphysics** menu. In that list, a bullet indicates the current application mode. To open a settings dialog box for another application mode, make it current by choosing it from the **Multiphysics** menu. You can now open the physics settings dialog boxes for this application mode. Any open physics settings dialog boxes for other application mode remain open and available for model specification. From the Model Tree, you can reach most of the dialog boxes on the **Physics** menu for all application modes in a model.

### **THE EQUATION SYSTEM VIEW**

The physics application modes and the application modes in the optional COMSOL modules provide built-in equations and the convenient specification of the physics that the application mode covers. Instead of entering PDE coefficients and general boundary coefficients, you select from typical material properties, sources, boundary conditions, and other physics settings. To find details on what these application modes provide, see the *COMSOL Multiphysics Modeling Guide* as well as the documentation that accompanies the optional modules.

COMSOL Multiphysics, however, always presents a model in terms of the underlying system of PDEs, the *equation system view*. In the **Physics** menu, point to **Equation System** and then open the dialog box for subdomain, boundary or point settings. You can also open these dialog boxes by Ctrl-clicking the corresponding selection mode button on the Main toolbar.

The equation system is the underlying representation of any model as a system of one or more PDEs in the coefficient or general forms. This representation is useful for several reasons:

- To let you introduce additional terms to a predefined physics mode
- To let you modify the equation, boundary conditions, and other settings from a predefined physics mode
- To add multiphysics couplings on the system level
- To check how COMSOL Multiphysics translates the equation and other settings in terms of physics

For more about the equation system views, see “Viewing and Modifying the Full Equation System” on page 218 and “Modifying Boundary Settings for the Equation System” on page 249 of the *COMSOL Multiphysics User’s Guide*.

# Example—Resistive Heating

For an example of a multiphysics model with thermal-electric couplings, look at resistive heating in a copper plate.

## *Introduction*

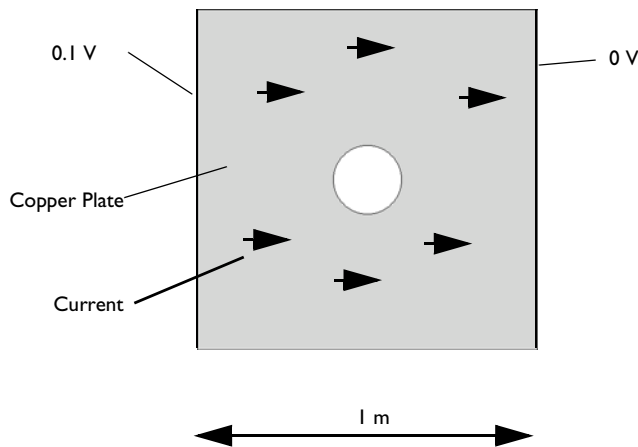
---

The material heats up when an electric current passes through it due to electric resistance. This is called resistive heating or Joule heating. There is also a coupling working in the opposite direction: the material's electric resistance varies with the temperature, increasing as the material heats up.

## *Model Definition*

---

Imagine a copper plate measuring  $1\text{ m} \times 1\text{ m}$  that also contains a small hole. The plate's thickness has no effect on the model. Suppose that you subject the plate to an electric potential difference across two opposite sides (all other sides are insulated). The potential difference induces a current that heats the plate.



*Figure 14-6: The model geometry and electric boundary conditions.*

In the 2D model you view the plate from above.

## *Modeling in COMSOL Multiphysics*

---

This example sets up the coupled thermal-electric analysis using the predefined multiphysics coupling for Joule Heating, which combines a Conductive Media DC application mode with a Heat Transfer by Conduction application mode. The potential distribution in the conductive media occurs almost instantly, but because unsteady heat transfer is a transient phenomenon, the full multiphysics model uses a transient analysis.

### **ADDITIONAL MULTIPHYSICS COUPLINGS**

By adding a structural mechanics application mode to this model you could represent a thermomechanical coupling with a body force proportional to the temperature gradient. For more information on thermomechanical multiphysics, see the model “Simulation of a Microrobot” on page 409 in the *COMSOL Multiphysics Model Library* and the documentation for the Structural Mechanics Module.

---

**Model Library path:** COMSOL\_Multiphysics/Multiphysics/  
resistive\_heating

---

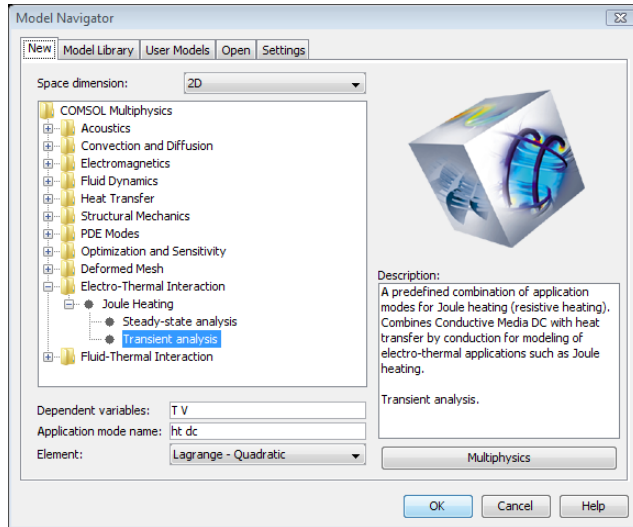
## *Modeling Using the Graphical User Interface*

---

### **MODEL NAVIGATOR**

- 1 On the **New** page, select **2D** from the **Space dimension** list.
- 2 In the list of application modes, open the **COMSOL Multiphysics>Electro-Thermal Interaction** folder and then the **Joule Heating** folder. Select **Transient analysis**.
- 3 Click **OK**.





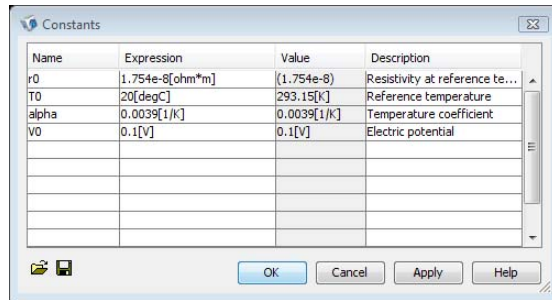
## OPTIONS AND SETTINGS

- 1 From the **Options** menu choose **Constants**.

2 Enter the following constant names, expressions, and (optionally) descriptions:

NAME	EXPRESSION	DESCRIPTION
r0	1.754e-8[ohm*m]	Resistivity at reference temperature
T0	20[degC]	Reference temperature
alpha	0.0039[1/K]	Temperature coefficient
V0	0.1[V]	Electric potential

Enter the constant's name in the **Name** edit field, then place its value and unit in the **Expression** field, and finally add a description in the **Description** edit field:



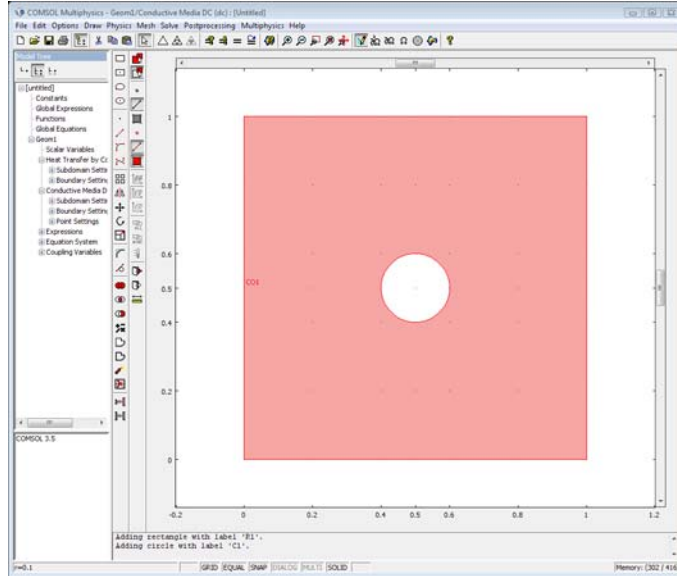
3 Click **OK**.

## GEOMETRY MODELING

Define the model geometry:

- 1 Shift-click the **Rectangle/Square (Centered)** toolbar button.
- 2 In the **Square** dialog box, click **OK** to use the default values and create a unit square with corners at (0, 0) and (1, 1).
- 3 Click the **Zoom Extents** toolbar button (on the Main toolbar).
- 4 Shift-click the **Ellipse/Circle (Centered)** toolbar button.
- 5 In the **Circle** dialog box, type 0.1 in the **Radius** edit field and then type 0.5 in the **x** and **y** edit fields in the **Position** area. Click **OK**.
- 6 To create a hole, select both geometry objects by pressing Ctrl+A.

7 Click the **Difference** button on the Draw toolbar.



## PHYSICS SETTINGS

The electric boundary conditions appear in Figure 14-6 on page 395. For the thermal boundary conditions, an air stream at 300 K (27 °C) cools the plate except on the thermally insulated upper and lower edges:

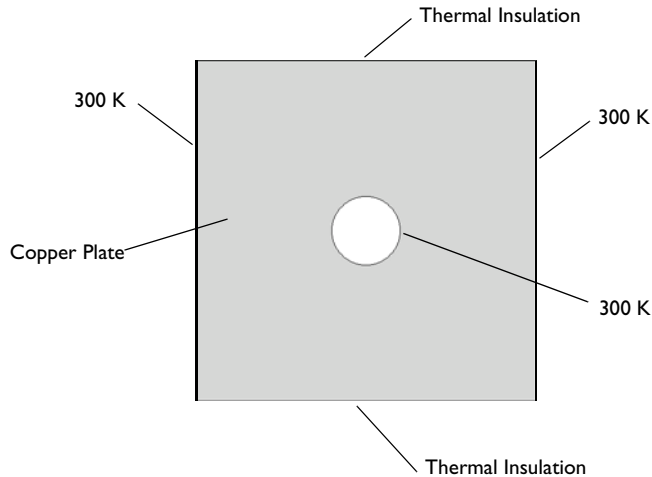


Figure 14-7: The model geometry and the thermal boundary conditions.

In Joule heating, the temperature increases due to the resistive heating from the electric current. The electric potential  $V$  is the solution variable in the Conductive Media DC application mode. The generated resistive heat  $Q$  is proportional to the square of the magnitude of the electric current density  $J$ . Current density, in turn, is proportional to the electric field, which equals the negative of the gradient of the potential  $V$ :

$$Q \propto |J|^2$$

The coefficient of proportionality is the electric resistivity  $\rho = 1/\sigma$ , which is also the reciprocal of the temperature-dependent electric conductivity  $\sigma = \sigma(T)$ . Combining these facts gives the fully coupled relation

$$Q = \frac{1}{\sigma} |J|^2 = \frac{1}{\sigma} |\sigma E|^2 = \sigma |\nabla V|^2$$

This resistive heating source term is directly available as the variable  $Q_{dc}$  ( $Q_{emdc}$  if you use the AC/DC Module) and is predefined as the source term in the heat transfer application mode when using the Joule Heating predefined multiphysics coupling.

Quantities other than  $\sigma$  also vary with temperature. For example, the thermal conductivity is temperature dependent, and a refined model would take this into account.

Over a range of temperatures the electric conductivity  $\sigma$  is a function of temperature  $T$  according to

$$\sigma = \frac{\sigma_0}{1 + \alpha(T - T_0)}$$

where  $\sigma_0$  is the conductivity at the reference temperature  $T_0$ .  $\alpha$  is the temperature coefficient of resistivity, which describes how the resistivity varies with temperature. A typical value for copper is 0.0039 per kelvin.

In the Conductive Media DC application mode you can specify the electric conductivity for Joule heating in terms of this equation. The predefined multiphysics coupling sets up this specification of sigma with the variable for temperature in the **T** edit field and default values suitable for copper. The only thing you have to add is the reference temperature,  $T_0$ .

*Boundary Conditions—Heat Transfer*

**1** Make sure that **Heat Transfer by Conduction (ht)** or **General Heat Transfer (htgh)** (if you use the Heat Transfer Module) is the selected application mode in the **Multiphysics** menu.

**2** Open the **Boundary Settings** dialog box from the **Physics** menu.

The thermal boundary conditions are:

SETTINGS	BOUNDARIES 1, 4-8	BOUNDARIES 2, 3
Type	Temperature	Thermal insulation
$T_0$	300	

**3** Select all boundaries.

**4** Select **Temperature** from the **Boundary condition** list.

**5** Type 300 in the  **$T_0$**  (temperature) edit field. This value corresponds to holding all boundaries at 300 K by positioning them in an air stream at room temperature.

**6** Select Boundaries 2 and 3, the top and bottom boundaries, and then select **Thermal insulation** from the **Boundary condition** list.

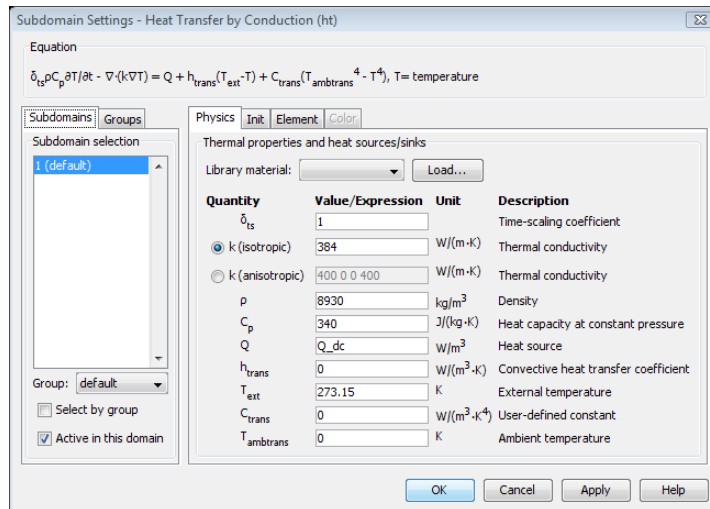
**7** Click **OK**.

### Subdomain Settings—Heat Transfer

The material properties for heat transfer are:

PROPERTY	VALUE
$\rho$	8930
$C_p$	340
k (isotropic)	384
Q	Q_dc

- 1 From the **Physics** menu choose **Subdomain Settings**.
- 2 Select Subdomain 1 from the **Subdomain selection** list.
- 3 Type 8930 in the **Density** edit field (the density of copper in  $\text{kg}/\text{m}^3$ ).
- 4 Type 340 in the **Heat capacity at constant pressure** edit field. The unit for heat capacity in this model is  $\text{J}/(\text{kg}\cdot\text{K})$ , the SI unit.
- 5 Type 384 in the **k (isotropic)** edit field for the thermal conductivity. This coefficient has the SI unit  $\text{W}/(\text{m}\cdot\text{K})$  and represents the material's ability to conduct heat per unit time.
- 6 The predefined setting in the **Heat source** edit field is Q\_dc (or Q\_emdc). This is a predefined variable for the resistive heating from the Conductive Media DC application mode and includes the temperature-dependent conductivity and the gradient of the electric potential.



### Initial Conditions—Heat Transfer

Also enter the initial value for the temperature, which is 300 K just as the boundary condition:

- 1 Click the **Init** tab in the **Subdomain Settings** dialog box.
- 2 Type 300 as the initial value in the edit field for  $T(t_0)$ .
- 3 Click **OK**.

### Application Mode Selection

To define the settings for the copper plate's electric properties, switch to the Conductive Media DC (dc) application mode, selecting it from the **Multiphysics** menu or using the Model Tree.

### Boundary Conditions—Conductive Media DC

- 1 Open the **Boundary Settings** dialog box by choosing **Boundary Settings** from the **Physics** menu. The boundary conditions for the Conductive Media DC application mode in this example are:

SETTINGS	BOUNDARY 1	BOUNDARY 4	BOUNDARIES 2, 3, 5–8
Type	Electric potential	Ground	Electric insulation
V	V0		

- 2 Press Ctrl+A to select all boundaries.
- 3 From the **Boundary condition** list select **Electric insulation**.  
Change the conditions on Boundaries 1 and 4:
- 4 From the **Boundary selection** list select Boundary 1, the left boundary.
- 5 From the **Boundary condition** list select **Electric potential** and type V0 in the  $V_0$  edit field.
- 6 Select Boundary 4.
- 7 From the **Boundary condition** list select **Ground**.
- 8 Click **OK**.

### Subdomain Settings—Conductive Media DC

- 1 From the **Physics** menu choose **Subdomain Settings**.
- 2 Select Subdomain 1 from the **Subdomain selection** list.
- 3 From the **Conductivity relation** list, select **Linear temperature relation**.
- 4 Make sure that the **T** edit field contains the field variable for temperature, T.
- 5 Type  $r_0$  in the  $\rho_0$  edit field for the resistivity at the reference temperature.

- 6 Type  $\alpha$  in the  $\alpha$  edit field for the temperature coefficient.
- 7 Type  $T_0$  in the  $T_0$  edit field for the reference temperature.

#### *Initial Conditions—Conductive Media DC*

- 1 In the **Subdomain Settings** dialog box, click the **Init** tab.
- 2 Type  $V_0*(1-x[1/m])$  in the edit field for  $V(t_0)$ . This expression means that the initial potential distribution varies linearly from  $V_0$  (0.1 V) at the left boundary ( $x = 0$ ) to 0 V at the right boundary ( $x = 1$ ). This is an initial condition that matches the boundary conditions. The reason for the multiplying  $x$  with the unit [1/m] is to make it dimensionless.
- 3 Click **OK**.

#### **MESH GENERATION**

Click the **Initialize Mesh** button on the Main toolbar to create a mesh using the default settings.

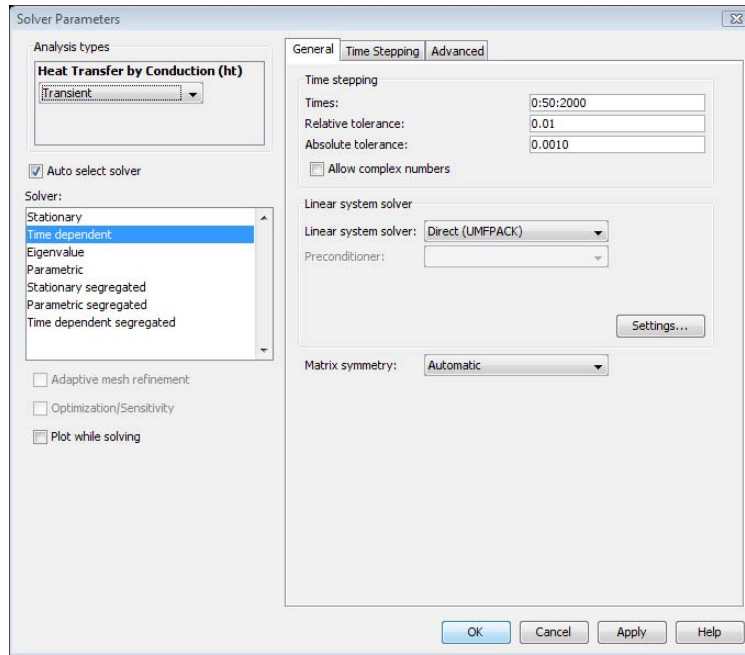
#### **COMPUTING THE SOLUTION**

The heat transfer in this model is a transient process, so the model uses a time-dependent solver for a transient analysis. First specify the simulation's time interval and the points in time to present solution data.

- 1 Open the **Solver Parameters** dialog box from the **Solve** menu.
- 2 Type  $0:50:2000$  in the **Times** edit field. Doing so produces a vector of 41 output times, linearly distributed every 50 seconds from 0 to 2000 seconds, for which the solution is available for postprocessing.



### 3 Click **OK**.



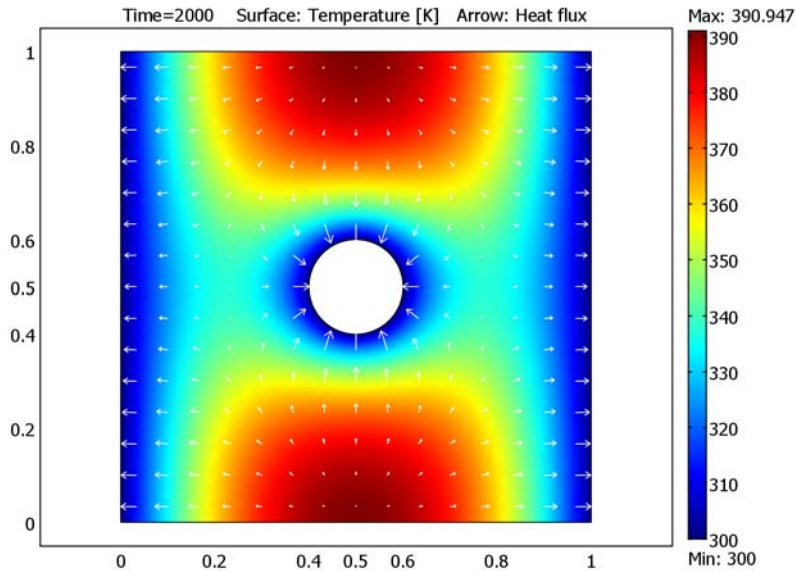
## COMPUTING THE SOLUTION

You can now run the transient analysis. To do so, click the **Solve** button on the Main toolbar.

## POSTPROCESSING AND VISUALIZATION

The default plot from this analysis shows the temperature or the potential distribution.

## Visualizing Heat Flux Using Arrow Plots



A quantity that is interesting to animate is the heat flux:

- 1 Open the **Plot Parameters** dialog box from the **Postprocessing** menu.
- 2 Click the **Arrow** tab.
- 3 Select the **Arrow plot** check box.
- 4 Select **Heat Transfer by Conduction (ht)>Heat flux** from the **Predefined quantities** list.
- 5 Click the **Color** button and select a color for the arrows, for example, white.
- 6 Click **OK**.

### Checking Heat Transfer Dynamics

Use a plot of temperature over time to determine if the time span for the simulation is sufficient to reach steady state:

- 1 From the **Postprocessing** menu choose **Cross-Section Plot Parameters**.
- 2 Make sure to select all time steps in the **Solutions to use** list on the **General** page.
- 3 Click the **Point plot** button.
- 4 Click the **Point** tab.
- 5 Select **Heat Transfer by Conduction (ht)>Temperature** in the **Predefined quantities** list.

6 In the **Coordinates** area, type 0.5 in the **x** edit field and 0.75 in the **y** edit field.

7 Click **OK**.

The plot of temperature as a function of time shows that it has reached a steady state. The temperature hardly increases at all if you extend the simulation over a longer time span.

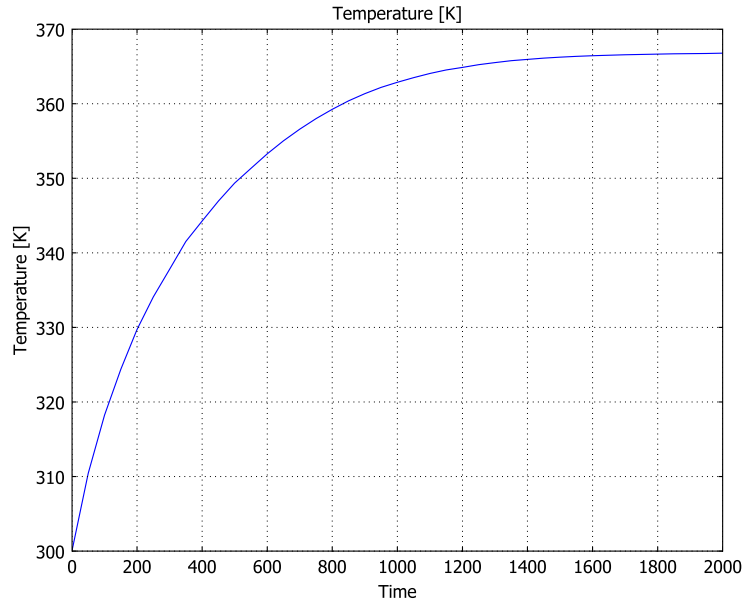


Figure 14-8: Heat transfer dynamics visualized as temperature versus time.

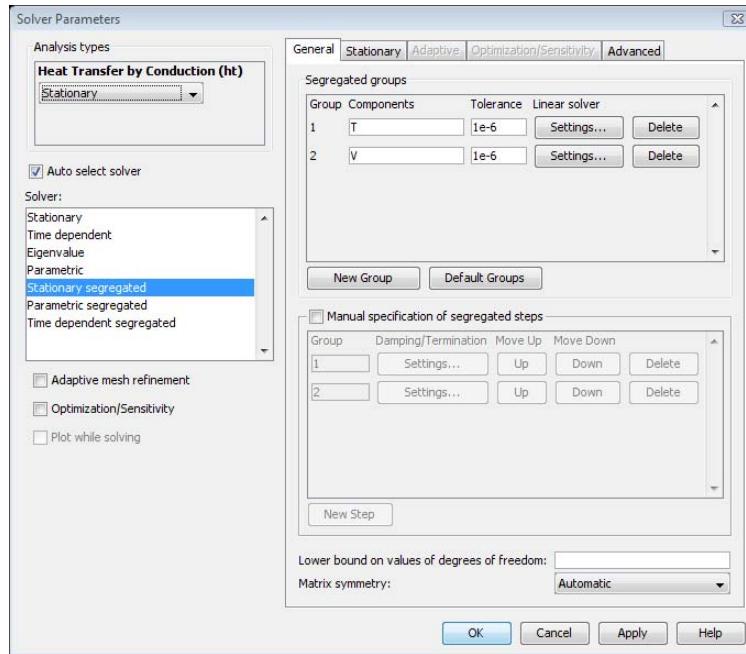
### COMPUTING THE SOLUTION USING A SEGREGATED STATIONARY SOLVER

You can also solve this model for the steady-state condition. There is no problem solving this coupled multiphysics model using the standard stationary solver because the model is small. For large models, a segregated solver approach can be beneficial. This example shows how to solve a multiphysics model using the segregated stationary solver. You continue with the Resistive Heating model, adding the following steps:

- 1 From the **Solve** menu, choose **Solver Parameters**.
- 2 Select **Stationary** from the **Analysis** list.
- 3 Select **Stationary segregated** from the **Solver** list. COMSOL Multiphysics has two predefined groups, one for each application mode (the temperature  $T$  and the electric potential  $V$ ). All you need to do is to specify the tolerances so that the

segregated solver scheme converges to the same solution as the standard stationary solver. Use the default linear solver settings (the direct solver UMFPACK).

- 4 Type  $1e-6$  in the **Tolerance** edit field for both groups (Group 1 for  $T$  and Group 2 for  $V$ ). The following figure shows the settings for the stationary segregated solver.



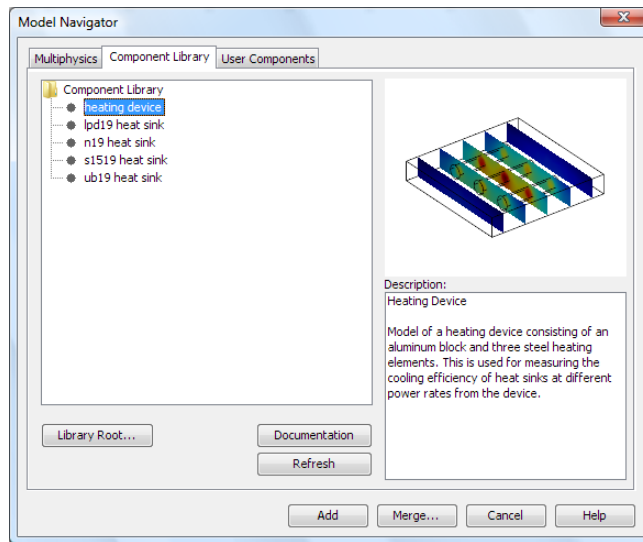
- 5 Click **OK**.
- 6 Click the **Solve** button on the Main toolbar.

# Working with Components

A multiphysics model can consist of one or several components. Each component can have material data and physics. To build a model consisting of several components you can build one component at a time and save it in a Model MPH-file. When you have all the components you can start to put them together by merging them with or adding them to the current model.

## *The Component Library*

To store Model MPH-files for use as components, you can use the Component Library. To open the Component Library, choose **File>Open Component Library**, or open the **Model Navigator** from the **Multiphysics** menu. Click the **Component Library** tab to see the components shipped with COMSOL Multiphysics.



*Figure 14-9: The Component Library in the Model Navigator.*

The **User Components** page is available for selecting those components you have created. Click the **Library Root** button to specify the root directory for your own component libraries.

## Adding Components

---

To add a component to a model on which you are working, select **Add Component** from the **File** menu; alternately, select the component you want to add in the Component Library and click the **Add** button. You can also add a component from the Model Tree when you choose the Detail and Inspect views. Right-click the top node with the model's name, and then select **Add Component**.

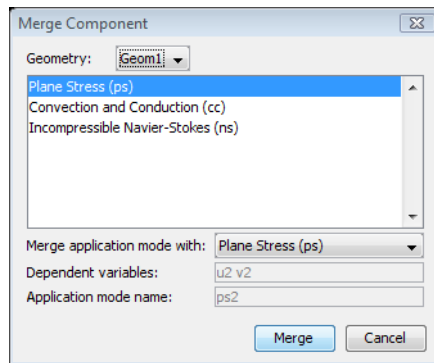
COMSOL Multiphysics creates a new geometry page and adds the component to this geometry. If the component has several geometries, the software adds all the geometries to the model. Adding a component includes the component's physics and mesh but not the solution.

## Merging Components

---

To merge a component to the model on which you are working, either select **Merge Component** from the **File** menu or select the component you want to merge in the **Component Library** and click the **Merge** button. You can also add a component from the Model Tree. Right-click the geometry node where you want to merge a component with the current mode, and then select **Merge Component**.

When merging a component the **Merge Component** dialog box appears.



In the **Geometry** list you can find the geometries in the component. Select the geometry you want to merge into the model. Clicking **Merge** merges the geometry objects in the selected geometry into the current geometry in your model. The resulting geometry becomes an assembly.

When you select a geometry in the dialog box, the application modes in this geometry appear in a list. If application modes of the same type as the selected one already exist

in the model, you can choose to merge the component's application mode with one of the application modes in the model. Select the model's application mode from the **Merge application mode with** list. The resulting application mode then includes the physics settings for the component.

If you want the component application mode to exist as a new application mode, select **None** in the **Merge application mode with** list. If you do not want to merge the application modes, you must make sure that the application mode names are unique. The program suggests unique names for the dependent variables and the application mode in the corresponding edit fields. If there are no application modes to merge with, or if **None** is selected, COMSOL Multiphysics adds a new application mode to the model. This application mode is only active in the domains belonging to the component.

When you have application modes in the model that the software does not merge with application modes from the component, they are active only in the domains that belonged to the model before the merge.

Because of geometry changes the merge does not include the mesh and the solution. Also, coupling variables that have other destinations than global are not merged.

For an example of how to work with components and merging them into a model, see “Heat-sink Experiments Using the Component Library” on page 309 of the *COMSOL Multiphysics Model Library*.





## Using Assemblies

This chapter explains how to model using assemblies where the geometry consists of separate parts. There are special considerations and possibilities during meshing as well as setting up the boundary conditions for an assembly model.

# Creating Assemblies

## *The Analyzed Geometry—Assembly vs. Composite Geometry*

---

When you leave Draw mode COMSOL Multiphysics analyzes the geometry to find all subdomains, boundaries, edges, and points in the geometry. When doing this COMSOL Multiphysics creates an *analyzed geometry* that you apply the physics on. You can choose between two types of analyzed geometries:

- A *composite geometry*. The analyzed geometry object that the software creates is a single object (that typically consists of several subdomains), which effectively is the same object you would get if you select all objects in Draw mode and create the union of them. This is the default type of analyzed geometries, for which COMSOL Multiphysics automatically connects the geometry, mesh, and physics at the interior interfaces.
- An *assembly*. In this case you get an analyzed geometry that consists of several parts (disjoint composite geometries), each object in Draw mode becoming a part of the assembly. For an assembly, you must manually connect the geometry, mesh, and physics at the part interfaces.

If there are no problems using a composite geometry this is usually preferred because it automatically provides continuity in the physical fields at part interfaces and material discontinuities as well as accurate solutions and conforming mesh elements and nodes at part interfaces. There are some cases, however, when an assembly is the best choice:

- Sometimes the software fails to create a composite geometry. In those cases you can choose to use an assembly. It is a simpler operation to create an assembly, and it often succeeds even if creating a composite geometry fails.
- Each part in an assembly gets a separate mesh. This can make it easier to mesh a complicated geometry, where the mesh generator creates a too dense mesh or even fails to create a mesh for the composite geometry. Bear in mind that having separate meshes means that the meshes might not be compatible at the two sides of a border between two parts. This can give you a less accurate solution.
- Assemblies make it possible to apply boundary conditions where the dependent variable is discontinuous at the interface between two parts. One example is a contact resistance boundary condition for electric currents where the electric potential is discontinuous.

To choose between using a composite geometry and using an assembly, toggle the **Use Assembly** menu item on the **Draw** menu.

## Creating Pairs

---

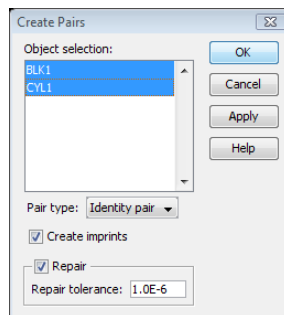
For the physics application modes to be able to connect the physics between the parts in the assembly you need to create *identity pairs*. The physics application modes use the pairs when defining their equations. An identity pair consists of two sets of domains: one set with domains from one part and a second set with domains in the other part. The domains in the pairs are those which are in contact with each other. Each pair consists of boundaries, edges, or points, but never a mixture of domain types. For more details about identity pairs, see the section “Identity Pairs” on page 418.

When you create the pairs, you can choose to make *imprints* of one geometry in another. If the boundaries of the two parts which are in contact with each other are of different sizes, COMSOL Multiphysics can create an imprint of the smaller boundary on the larger boundary. In 2D this inserts points on the larger boundary, and in 3D this creates a curve on the larger boundary. The advantage of making imprints is that the two sides of the pair match. When you mesh the parts, the meshes match along the border of the pair of boundaries. This provides a more accurate solution than in the case without the imprint, where the meshes do not match.

To create the pairs either use the **Create Pairs** dialog box on the **Draw** menu or the **Create Pairs and Imprints** button and the **Create Pairs** button on the Draw toolbar.

### EXAMPLE OF CREATING PAIRS

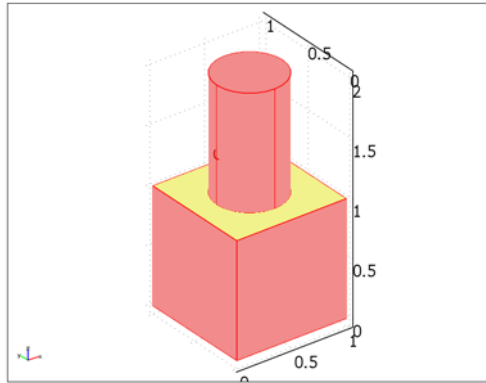
This example uses a cylinder placed on top of a cube. To create a pair between them open the **Create Pairs** dialog box by choosing **Create Pairs** from the **Draw** menu.



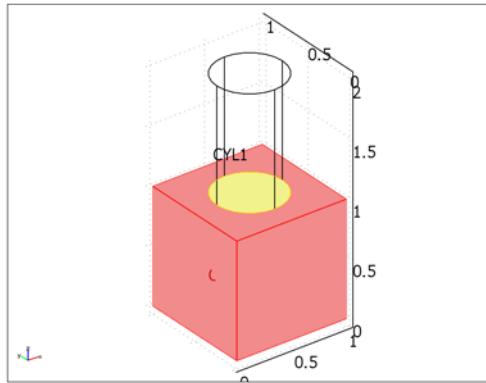
The **Create imprints** check box controls whether to create imprints or not. In the list select which objects to create pairs between. In this case there are only two objects, but when there are more than two, the software tries to create pairs between all pairs of the selected objects. For each pair of objects that touch each other the software creates identity pairs, usually one boundary pair, one edge pair, and one point pair.

If you create pairs between the cube and the cylinder without imprints, the rectangular surface of the cube and the circular end of the cylinder become an identity pair.

In the following figure the boundaries that belong to the boundary pair have a yellow color.



If you instead create pairs and imprints, the software creates a circular curve on the cube's surface. This circular surface on the cube and the cylinder surface become a pair. In the following figure you can see this circle.



If your license includes the Structural Mechanics Module or the MEMS Module you can also create *contact pairs* using the **Create Pairs** dialog box. To do this select **Contact pair** in the **Pair type** list. This list is absent in the dialog box if you do not have any of these two modules. Contact pairs are used when modeling structural contact. You can read about this topic in the *Structural Mechanics Module User's Guide* and the *MEMS Module User's Guide*.

### **CONTROLLING THE REPAIR TOLERANCE**

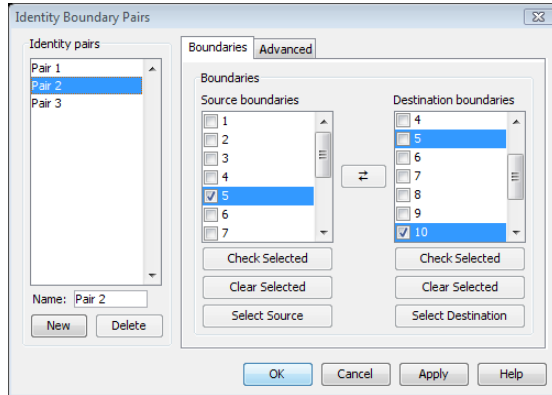
When creating pairs, the software ignores gaps that are smaller than a certain tolerance. You can change this value in the **Repair tolerance** edit field. This value is relative to the overall dimensions of the geometry. For example, if the dimensions are in meters, the default repair tolerance of  $10^{-6}$  makes the software ignore gaps that are smaller than about a micrometer ( $10^{-6}$  m). If your geometry has larger defects, you might need to increase this tolerance. To resolve extremely small gaps, you might need to decrease the repair tolerance. To turn off the repair functionality, clear the **Repair** check box. This is equivalent to using  $10^{-10}$  as repair tolerance. Note that the repair tolerance in the **Create Pairs** dialog box is used also when you click the **Create Pairs and Imprints** button or the **Create Pairs** button on the Draw toolbar.

# Identity Pairs

When making models with assemblies you need to define *identity pairs*, which define where the parts of the assemblies are connected. The application modes use the pairs to put constraints on the equations such that the solution becomes continuous across the border between the parts. An identity pair consists of two sets of domains, either two sets of boundaries, two sets of edges, or two sets of points. Such pairs are called identity boundary pairs, identity edge pairs, and identity point pairs, respectively. The two sets of domains are the *source domains* and the *destination domains*. The application modes put a constraint on the dependent variables on the destination domains, forcing them to be equal to the value on the source domains. When the source and destination do not match each other completely you obtain the best numerical result if you let the destination be the smaller of the two. When creating pairs using either the **Create Pairs** dialog box or any of the **Create Pairs and Imprints** button or the **Create Pairs** button on the Draw toolbar, the smaller side always becomes the destination. The easiest way to create the identity pairs is to use the dialog box or the toolbar buttons. For details see the section “Creating Pairs” on page 415; it describes how to manually define the pairs, which becomes necessary if the automatic generation fails, or if you need to adjust the pairs that the automatic generation created.

## The Identity Pair Dialog Box

To define identity pairs, choose any of the commands on the **Physics>Identity Pairs** menu. Doing so opens the dialog box for the identity pairs of the selected domain type, for example, the **Identity Boundary Pairs** dialog box.



Each pair has a name. The application modes use this name to refer to the pair. The names must be unique for each domain level, but pairs on different domain levels can have the same name. For example, two boundary pairs cannot have the name “Pair 1,” while there can be both a boundary pair and a point pair called “Pair 1.”

The two domain lists show the source and destination domains of the pair selected in the list on the left. The check boxes beside the domain numbers indicate which domains belong to the source and which to the destination.

Clicking the buttons **Check Selected** below the lists selects the check boxes of the domains highlighted in the list. This is equivalent to selecting the individual check boxes and is a quick way to select multiple check boxes. Clicking the button **Clear Selected** similarly clears the check boxes of the selected domains.

Use the buttons **Select Source** and **Select Destination** to select the source and destination domains in the main window and in the selection lists.

Clicking the arrow button between the selection lists interchanges the source and destination domains.

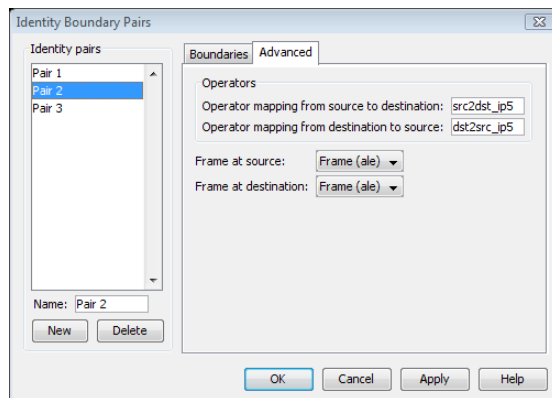
## TIPS FOR MODELING USING IDENTITY PAIRS

Note these aspects of the modeling procedure using identity pairs:

- When adding identity pairs, use the domain with the coarser mesh as the source domain and the domain with a finer mesh as the destination domain.
- When using identity pairs to connect boundaries or edges that do not have equivalent meshes, use the sparse null-space function, particularly in 3D. Normally the automatic selection takes care of this. To specify the sparse null-space function, open the **Solver Parameters** dialog box, click the **Advanced** tab, and select **Sparse** from the **Null-space function** list.

## COUPLING OPERATORS

On the **Advanced** page in the **Identity Boundary Pairs**, **Identity Edge Pairs**, and **Identity Point Pairs** dialog box you can define the names of the pair's coupling operators. A *coupling operator* evaluates its argument on one side of the pair and makes the result available on the other side.



In the previous figure you can see two operators: `src2dst_ip5`, mapping from the source of the pair to the destination, and `dst2src_ip5`, mapping in the other direction. For example, if  $u$  is a variable on the source side then you can use the expression `src2dst_ip5(u)` on the destination side.

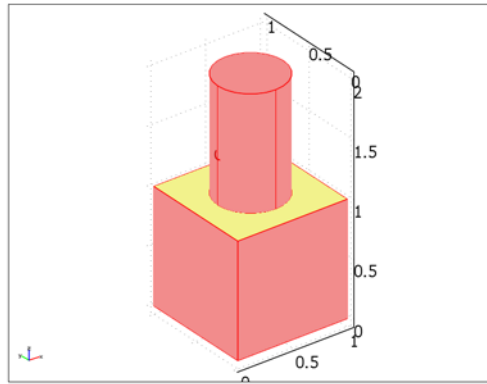
When creating a pair COMSOL Multiphysics automatically defines the operators and give them a name. The names have to be unique within the whole model.

The application modes use the operators to define the constraint which forces the dependent variables to be continuous across the border between the parts of the



assembly. For example, if  $u$  is a dependent variable, the application modes define the constraint `src2dst_ip5(u) = u` on the destination side of the pair.

You can also use the operators in logical expressions. The operator name without an argument evaluates to true in the points which have a corresponding source point in the same position. As an example consider the cube and cylinder in Figure 15-1 below. Assume that the rectangle on top of the cube is the source of a pair and that the bottom cylinder surface is the destination. Further assume that the operator `src2dst` maps from the source to the destination. Then you can use the expression `if(src2dst, 1, 2)` on the source side (the cube). It evaluates to 1 in the points on the surface touching the cylinder and to 2 in the other points.



*Figure 15-1: A cube and a cylinder which are two parts of an assembly. The cube's upper surface is the source boundary of an identity pair and the cylinder's bottom surface is the destination boundary of the pair.*

#### **USING FRAMES FOR PAIRS**

If there is more than one frame in a model you can choose on which frame to evaluate the expressions when using coupling operators. If you choose two different frames for the source and destination, you can simulate that the source and destination are translated or rotated relative to each other.

The panel where you can select frames does not appear when there are no extra frames in the model.

## *Selection Colors for Pairs*

---

When you select a domain it is normally highlighted in red. In 3D selected domain can also be blue and green for confirmed selections (see “Object Selection Methods in 3D” on page 127 of the *COMSOL Multiphysics User’s Guide*). In addition, pairs have their own selection colors.

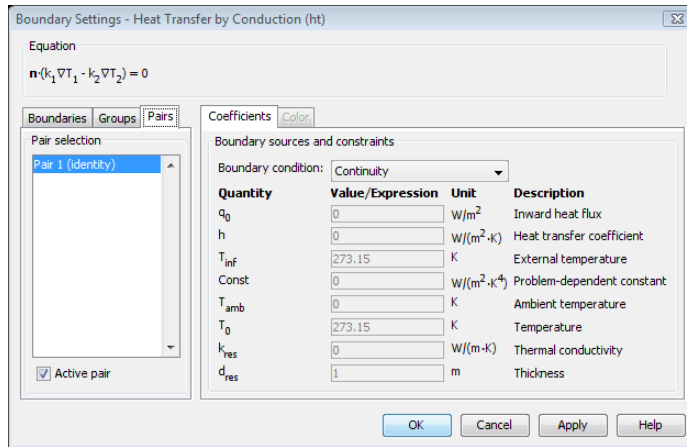
In Draw mode the domains belonging to a pair have a yellow color. This is true whether they are selected or not.

In the Subdomain, Boundary, Edge, and Point selection modes the pair domains have three possible selection colors. When you select a source domain it becomes highlighted in cyan. A destination domain which you select becomes highlighted in magenta. If a domain is a source domain to one pair and a destination domain to another pair it is highlighted in yellow. These specific selection colors for pairs override the red selection color but not the green and blue colors for confirmed selections.

You can deactivate the pair colors and use only the red selection color. To do so open the **Preferences** dialog box, and on the **Visualization** page clear the **Pair colors** check box. This also turns off the yellow color for the pairs in Draw mode.

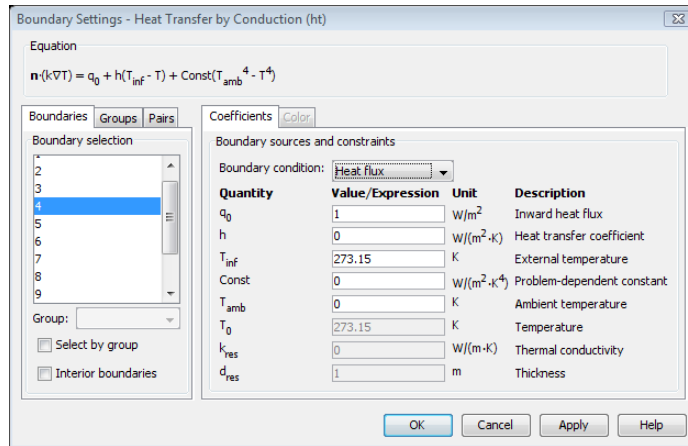
# Specifying Physics Settings on Pairs

The application modes use the identity pairs described in the previous chapter to define the boundary conditions at the border between the domains in the pair. The default boundary condition ensures continuity of the dependent variables across this border, but you can specify other boundary conditions. To do this, use the **Boundary Settings**, **Edge Settings**, and **Point Settings** dialog boxes. Whenever there are pairs in a model, a **Pairs** page appears in the dialog box (see the following figure).



The boundary condition that you apply to a pair is active everywhere where the two sides of the pair are in contact. But you can also set boundary conditions on the individual boundaries of the pair. Do so on the **Boundaries** page. These settings are active everywhere where the domains of the pair are not in contact with each other. To clarify how this works, consider the example of a cube with a cylinder on top of it in Figure 15-1 on page 421. Assume that the cube's upper surface is Boundary 4 and the cylinder's bottom surface is Boundary 9. The boundary condition on the pair applies to the circular area where the two surfaces are in contact. You specify the boundary

condition for the rest of Boundary 4 on the **Boundaries** page. In the following figure this boundary condition is a heat flux boundary condition.



If you specify a boundary condition at Boundary 9 in this list, this boundary condition has no effect. This is because all of Boundary 9 is in contact with Boundary 4 and therefore the boundary condition of the pair applies everywhere on this boundary.

### *Controlling the Use of Pairs*

On the **Pairs** page you find the **Active pair** check box. Clearing this check box makes the pair inactive for this application mode. In this case the application mode does not use the boundary condition on the pair but instead uses the boundary conditions on the individual boundaries that you specify on the **Boundaries** page. When a pair is not active the application mode does not ensure that the dependent variables are continuous across the border between the parts. Instead the boundaries on the two sides of the pair behave like exterior boundaries, and the parts are completely insulated from each other.

Pairs also become inactive if you have deactivated the application mode in one part of the geometry such that one or both sides of the pair belongs to the inactive part. In that case the active side of the pair is an exterior boundary, and the settings on the **Boundaries** page apply.

## *Edge and Point Pairs*

---

You specify the settings for edge and point pairs in the same way as for boundary pairs. But most application modes do not use the edge and point pairs. The application modes that define their equations on the subdomain level use boundary pairs only. The reason is that because the setting on the boundary pair ensures that the dependent variables are continuous between the boundaries, this also ensures that they are continuous between the edges and points adjacent to the boundaries in contact. Putting constraints on the edges and points is therefore unnecessary.

The application modes which define their equations on the boundary level use the pairs on edges in 3D or on points in 2D. The dependent variables of these application modes need a constraint on the edges (or points) to be continuous.

Similarly, the application modes that define their equations on the edge level use the point pairs.

## *Slit Boundary Conditions*

---

On the borders between pairs in an assembly, you can specify special “slit boundary conditions,” which represent a discontinuity in the field due to, for example, contact resistance. The following application modes include a boundary condition of this type:

TABLE 15-1: SLIT BOUNDARY CONDITIONS

APPLICATION MODE	BOUNDARY CONDITION	PHYSICAL QUANTITY
Conductive Media DC	Contact resistance	Electric conductivity
Electrostatics	Thin highly capacitive layer	Permittivity
Diffusion, Convection and Diffusion, Electrokinetic Flow	Thin boundary layer	Diffusion coefficient
Heat Transfer by Conduction, Convection and Conduction, General Heat Transfer	Thin thermally resistive layer	Thermal conductivity
Magnetostatics, No Currents	Thin low permeability gap	Permeability

These boundary conditions are available in the **Boundary Settings** dialog box by clicking the **Pairs** tab and selecting the pairs where you want to apply the boundary condition. In addition to the physical quantity listed in Table 15-1, you must also enter the thickness of the layer or gap. For an example model, see “Example—Thin-Film Resistance” on page 430.

---

**Note:** The General Heat Transfer application mode is available only in the Heat Transfer Module, and the Magnetostatics, No Currents application mode is available only in the AC/DC Module. The Electrokinetic Flow application mode is available only in the MEMS Module and the Chemical Engineering Module.

---

# Meshing Assemblies

A geometry object that is to be meshed is classified as either a *composite geometry* or an *assembly*. For more information on the two geometry types see “The Analyzed Geometry—Assembly vs. Composite Geometry” on page 414.

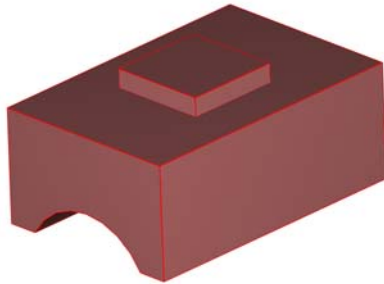
When meshing an assembly, the mesh generator creates meshes for the individual parts of the assembly independently of each other. This means that a geometry object of assembly type often gives you more freedom when determining the meshing strategy than the corresponding composite geometry. Furthermore, in most cases, an assembly generates a mesh with fewer elements than the corresponding composite geometry.

Whether you create an assembly with or without imprints also affects the meshing process. An assembly created without imprints gives you even more freedom when determining the meshing strategy than the corresponding assembly with imprints.

## *Example—Meshing a Composite Geometry or an Assembly*

---

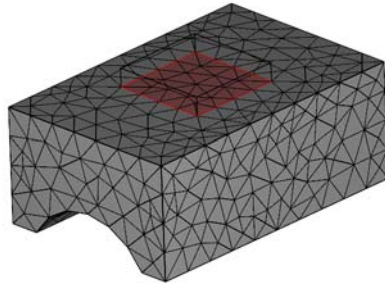
To show the different meshing possibilities for a geometry object of composite geometry type and assembly type (with and without imprints), this example uses the following geometry object consisting of two subdomains.



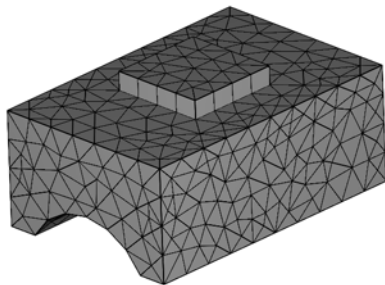
### **COMPOSITE GEOMETRY**

If the geometry object is a composite geometry, there is only one face connecting the two subdomains. This face, common to both subdomains, is called an *interior boundary* of the geometry object. Due to topology reasons of the lower subdomain

(that is, the connectivity of the faces about the subdomain) you must use the free mesher when meshing the lower subdomain. This means that the interior boundary face contains triangular boundary elements (see the following figure).



When meshing the upper subdomain you can use the free mesher or the swept mesher. However, if you use the swept mesher for this subdomain the interior boundary face must be used as the source face or the target face for the sweep. Hence, it is not possible to create a swept hexahedral mesh on the upper subdomain. The next figure shows a swept prism mesh on the upper subdomain.

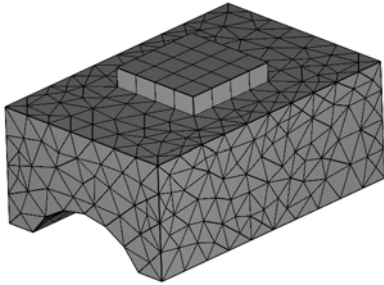


#### **ASSEMBLY WITH IMPRINTS**

If you model the geometry object as an assembly with imprints, COMSOL Multiphysics creates two separate faces of the same shape, belonging to each part, at the border between the parts. The meshes for these faces are independent of each other. Due to the imprint on the lower part, it is only possible to create a free mesh on

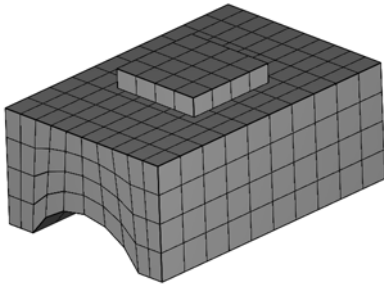


this part. However, because you can create a mesh for the upper part independently of the lower part you can create a swept hexahedral mesh on the upper subdomain.



#### **ASSEMBLY WITHOUT IMPRINTS**

If you form an assembly without imprints you can freely mesh the two parts (subdomains) independently of each other. Because there is no imprint of the upper part on the lower part, it is now possible to create a swept mesh on the lower part as well. In the next figure, the mesh generator creates a swept hexahedral mesh on both parts of the assembly.



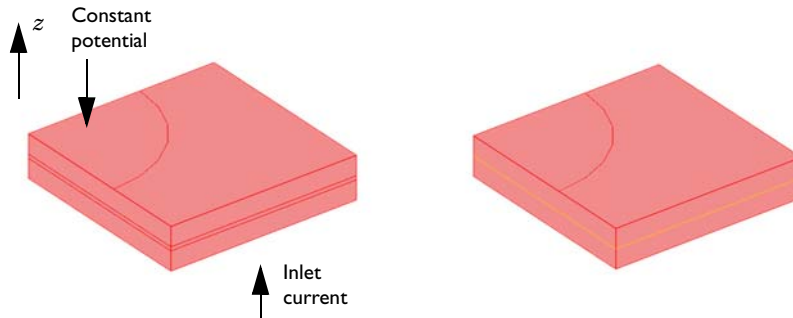
# Example—Thin-Film Resistance

When modeling transport by diffusion or conduction in thin layers, large differences in dimensions of the different subdomains are common. If the model has a sandwich structure, you can replace the thinnest layers with a thin-layer approximation, provided that the difference in thickness is large.

## *Model Definition*

---

This study explains the principle of the thin-layer approximation in direct current conduction problems. A comparison of a structure with three subdomains to a simplified model that replaces the domain in the middle with a thin-layer approximation shows the benefit of this approach (see Figure 15-2).



*Figure 15-2: Exact domain description (left) and approximation (right). The current flows from the base plate to the circular plate on the upper surface of the device.*

Equation 15-1 below describes the current balance in all three subdomains in the real sandwich structure:

$$\nabla \cdot (-\sigma \nabla V) = 0 \quad (15-1)$$

In this equation,  $\sigma$  represents the conductivity and  $V$  the electric potential. In this case, there is a substantial difference in conductivity between the thin and thicker layers of the structure. The boundary conditions include a current inlet in the base plate of the device and a constant potential at the upper circular boundary (see Figure 15-2). All other boundaries are insulated.

The simplified model is based on the assumption that the components of the current density vector in the  $x$  and  $y$  directions are small and that the dominating transport

through the thin structure is obtained in the  $z$  direction. For the middle layer, this implies that you can approximate Equation 15-1 by the one-dimensional equation

$$-\sigma \frac{d^2 V}{dz^2} = 0 \quad (15-2)$$

It is possible to solve this equation analytically if the potential is given at the lower and upper surfaces of the middle layer:

$$V_{\delta=0} = V_1 \quad (15-3)$$

$$V_{\delta=\delta_1} = V_2 \quad (15-4)$$

You can integrate Equation 15-2 analytically to give:

$$V = az + b \quad (15-5)$$

where  $a$  and  $b$  are integration constants. If you arbitrarily place  $z = 0$  at the lower boundary of the middle layer, you get the constants  $a$  and  $b$  from the boundary conditions in Equation 15-3 and Equation 15-4:

$$V_1 = b \quad (15-6)$$

$$V_2 = a\delta + b \quad (15-7)$$

This gives:

$$b = V_1 \quad (15-8)$$

$$a = \frac{V_2 - V_1}{\delta} \quad (15-9)$$

The resulting equation for the potential is thus

$$V = \left( \frac{V_2 - V_1}{\delta} \right) z + V_1 \quad (15-10)$$

The current density is defined as

$$J_z = -\sigma \frac{dV}{dz} \quad (15-11)$$

Combining Equation 15-10 and Equation 15-11 gives

$$J_z = -\sigma \left( \frac{V_2 - V_1}{\delta} \right) \quad (15-12)$$

In the thin-film approximation the potential is discontinuous at the film boundary. Using an assembly it is possible to model such a potential. The Conductive Media DC application mode has a contact resistance boundary condition available at pairs, which uses the thin-film approximation.

It is also possible to derive the expression for the current density in Equation 15-12 by approximating the gradient using the potential difference over the thin layer. This example includes the previous tedious derivation to show that this is exactly what you obtain from the solution of Equation 15-2.

The approximation presented in this example is not limited to direct current problems: You can also use it for modeling of diffusion, heat conduction, flow through porous media using Darcy's law, and other types of physics that the divergence of a gradient flux describes.

In general, the application of this simplification is appropriate in cases where the differences in thickness are so large that the mesh generator cannot even mesh the domain. In some cases, the mesh generator might be able to mesh the domain but then creates a very large number of elements.

### *Results*

---

Figure 15-3 shows a comparison between the exact solution of the problem using three conductive layers and the thin-film approximation. The comparison reveals an excellent agreement in the potential and current distribution despite that the middle film in this study is relatively thick. The approximation becomes even more accurate as the film thickness between the upper and lower subdomain decreases.

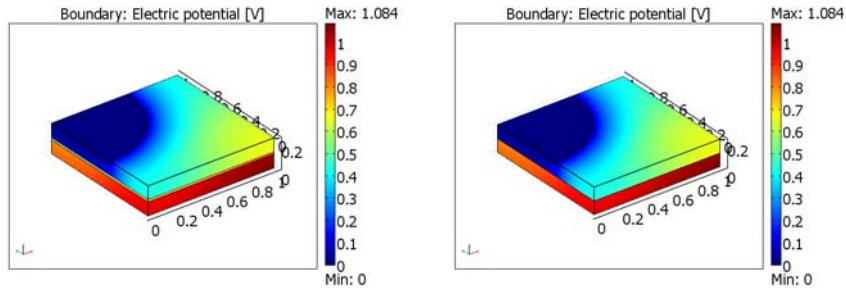


Figure 15-3: Potential distribution in the modeled device. The value of the potential loss over the device at a current of 0.3 A is almost identical in the two models: the full model (left) and thin-film approximation (right).

Figure 15-4 shows a cross-section plot of the potential through the structure's center for the full model and for the approximation. The plots show the excellent agreement obtained between the two models.

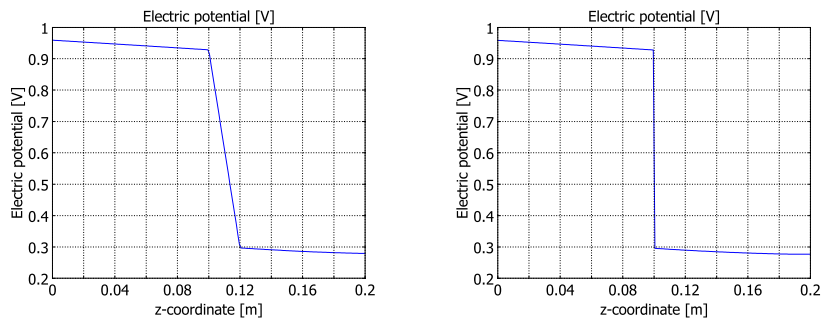


Figure 15-4: Potential distribution along the  $z$  direction in the middle of the device. Solution for the full model (left) and for the thin-film approximation (right).

---

**Model Library path:** COMSOL\_Multiphysics/Electromagnetics/  
thin\_film\_resistance

---

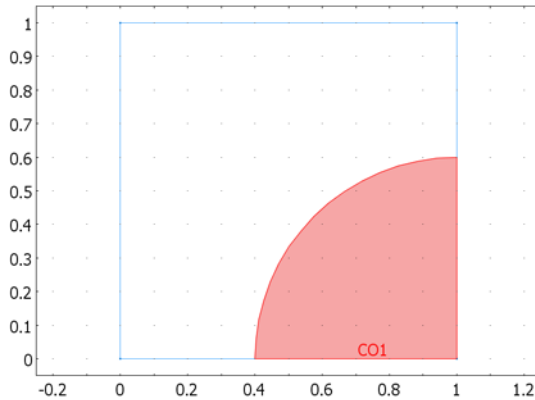
#### **MODEL NAVIGATOR**

- 1 Start COMSOL Multiphysics. Under the **New** tab select **3D** in the **Space dimension** list.
- 2 From the list of application modes, select **Conductive Media DC**, which you find under **COMSOL Multiphysics>Electromagnetics**.
- 3 Click **OK**.

#### **GEOMETRY MODELING**

- 1 Draw the bottom block by selecting **Draw>Block**. Set the thickness of the block to 0.1 in the  $z$  direction (this is set in the **Length** frame) and leave the other settings as they are. Click **OK**.
- 2 Draw the top block in the same way except for the **Axis base point**, which should be at  $x = 0$ ,  $y = 0$ , and  $z = 0.1$ .
- 3 Click the **Zoom Extents** button on the Main toolbar to get a better view of the geometry.
- 4 Now go to **Draw>Work-Plane Settings** and click **OK** to create a work plane with the default settings. A work plane is created under the new **Geom2** tab, while the original 3D geometry remains under the **Geom1** tab.
- 5 Now go to the work plane by clicking on the **Geom2** tab. Click **Zoom Extents** to see the outlines of the 3D geometry from **Geom1**.
- 6 Draw a circle centered at (1, 0) with a radius of 0.6. This is done by selecting **Draw>Specify Objects>Circle**. In the **Size** area set the **Radius** to 0.6, and in the **Position** area set **Base** to **Center** and the coordinate **x** to 1 and **y** to 0. Click **OK**.
- 7 Draw a square by clicking the **Rectangle/Square** button from the Draw toolbar on the left and click on the point (0, 1) and then (1, 0).
- 8 Create an object by finding the intersection of the square and the circle. This is done by selecting both objects (press Ctrl+A) and then clicking on the **Intersection** tool

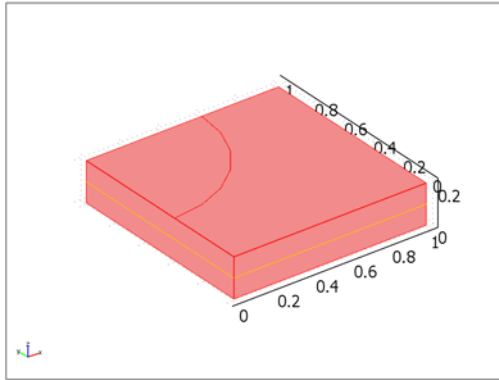
on the Draw toolbar on the left. The 2D geometry in **Geom2** should now look as in the figure below:



*The work plane.*

- 9 Now embed the quarter-circle in the 3D geometry by selecting **Draw>Embed** and clicking **OK**.
- 10 Move the flat quarter-circle surface to the top corner of the structure by pressing Ctrl+X followed by Ctrl+V. For the displacements, type -1, 1, and 0.2 in the **x**, **y**, and **z** edit fields, respectively, then click **OK**.
- 11 Rotate the quarter circle by first selecting it and then clicking the **Rotate** button on the Draw toolbar on the left. Type 180 in the **Rotation angle** edit field (the rotation angle  $\alpha$  in degrees) and specify the point on the rotation axis to be (0, 1, 0) in the **Point on rotation axis** area. Click **OK**.
- 12 Select the embedded object EMB1 and the upper block BLK2 and click the **Coerce to Solid** button to combine the two objects to a single object.
- 13 Click the **Create Pairs** button on the Draw toolbar to create the pair joining the two objects.

The final geometry should look as in the figure below:



*The final 3D geometry.*

## PHYSICS SETTINGS

### Subdomain Settings

- 1 Select **Physics>Subdomain Settings**.
- 2 Select all subdomains.
- 3 In the  $\sigma$  edit field (**Electric conductivity**) type 1. Click **OK**.

### Boundary Conditions

- 1 Select **Physics>Boundary Settings** and set the following boundary conditions:

SETTINGS	BOUNDARY 3	BOUNDARY II	ALL OTHERS
Boundary condition	Inward current flow	Ground	Electric insulation
$J_n$	0.3		

- 2 Click the **Pairs** tab and select **Pair I**. From the **Boundary condition** list, select **Contact resistance**. In the  $\sigma$  edit field, enter the electric conductivity  $1e-2$  and in the **d** edit field enter the thickness 0.02. Click **OK**.

## COMPUTING THE SOLUTION

Click the **Solve** button on the Main toolbar to mesh and solve the model.

## POSTPROCESSING AND VISUALIZATION

Reproduce the right panel of Figure 15-3 by following these instructions:



- 1 Click the **Plot Parameters** button on the Main toolbar.
- 2 On the **General** page, clear the **Slice** check box and select the **Boundary** check box.
- 3 Click the **Boundary** tab.
- 4 From the **Predefined quantities** list, select **Conductive Media DC (dc)>Electric potential**.
- 5 Clear the **Smooth** check box.
- 6 Click **OK**.

To generate the plot in the right panel of Figure 15-4, proceed with the following steps:

- 1 From the **Postprocessing** menu, select **Cross-Section Plot Parameters**.
- 2 On the **Line/Extrusion** page, verify that the selection in the **Predefined quantities** list in the **y-axis data** area is **Electric potential**.
- 3 In the **x-axis data** area, click first the lower option button and then the **Expression** button.
- 4 In the **X-Axis Data** dialog box, type **z** in the **Expression** edit field. Click **OK**.
- 5 In the **Cross-section line data** area, enter the following settings:

x0	0.5	x1	0.5
y0	0.5	y1	0.5
z0	0	z1	0.2

- 6 Click **OK** to close the **Cross-Section Plot Parameters** dialog box and generate the plot.

### *Comparing the Thin-Film Approximation with the Full 3D Model*

You can easily reproduce the full 3D model that served as a reference model for comparison in this example. To do so, move the upper block, CO1, a distance 0.02 in the  $z$  direction, then add a thin block with the height 0.02 in between the top and bottom block. Here it is not necessary to use an assembly, so you can clear **Use Assembly** in the **Draw** menu. Set the conductivity to 0.01 in this thin middle subdomain (leave it at 1 in the top and bottom subdomains). Let the boundary conditions for the top and bottom of the thin layer be inactive, which means that the solution is continuous across the interfaces. Let the exterior boundary conditions be the same (that is, electric insulation), and compute the solution for this model.

It should now be apparent that the two solutions show good agreement and that the first approach consumes significantly less time and memory.

# Using Identity Conditions

To connect the fields of physics across different geometries, you can use identity conditions that make the physical quantities equal across the different parts. The identity boundary conditions and identity conditions on other domain types define a constraint that makes two quantities equal on two separate (but usually equally-shaped) domains in two different geometries where you do not need a coordinate transformation between the source and destination domains. That is, the source and the destination domains must lie in the same place in the coordinate space.

---

**Note:** Using an *assembly* and *identity pairs*, you can work with a geometry that consists of several parts without using multiple geometries and identity conditions.

---

## *Creating Identity Conditions*

---

When you do not need a transformation, you specify an identity condition in the same way as you specify a periodic condition:

- 1 Select the source geometry for the coupling variable.
- 2 In the **Options** menu, point to **Identity Conditions**, and then click on the domain type: **Identity Subdomain Conditions**, **Identity Boundary Conditions**, **Identity Edge Conditions** (3D only), or **Identity Point Conditions**.
- 3 In the dialog box for the identity conditions, start by specifying the source geometry on the **Source** page. Enter the expression that you wish to use in the identity coupling for the source. Refer to each constraint in the identity coupling by a name. A default name appears for the constraint that implements the identity coupling. To

change the default name of the constraint while working on the source page, type another name in the **Constraint name** column.

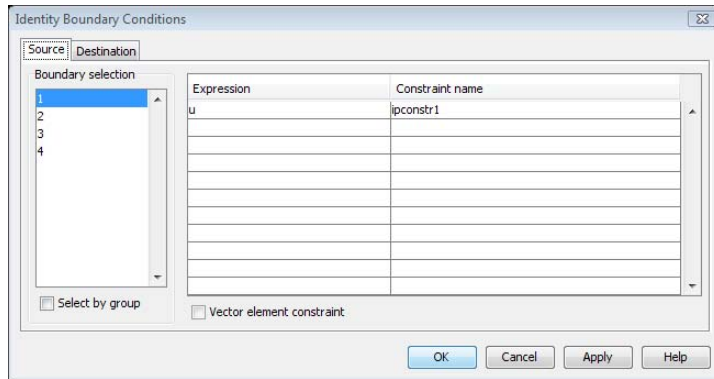


Figure 15-5: The Source page in the Identity Boundary Conditions dialog box.

- 4 Click the **Destination** tab to define the destination domains. Select the destination geometry in the **Geometry** list. Then select the constraint that you want to specify in the **Constraint name** list. Select the check boxes for the appropriate domains such as Boundary 4 in Figure 15-6, for example. Finally specify the expression that becomes equal to the expression on the source domain for the corresponding constraint.

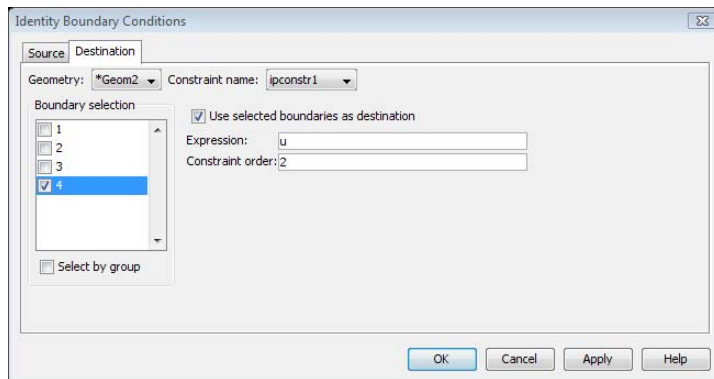


Figure 15-6: The Destination page in the Identity Boundary Conditions dialog box.

- 5 Click **OK** when you have defined all the identity conditions for the domain type that you are working with.

See the model example on the following pages for a description of how to combine two geometries with brick and prism meshes using an identity boundary condition.

## MODELING PROCEDURE WHEN USING IDENTITY CONDITIONS

Note the following aspects of the modeling procedure when using identity conditions:

- When adding identity conditions, use the domain with the coarser mesh as the source domain and the domain with a finer mesh as the destination domain.
- For an identity boundary condition, use a Neumann boundary condition for the boundaries on both the source and destination domains. The identity boundary conditions do not work if you use a Dirichlet boundary condition. You specify the boundary conditions in the **Boundary Settings** dialog box.
- If the source domain is larger than the destination domain and you would like to use a different type of boundary condition on the part of that boundary that is not connected to the destination boundary in the other domain, you can introduce a smaller domain that matches the destination domain. This might, however, limit the possibilities to use a mapped 2D mesh and then also a 3D mesh using brick elements. In that case, consider using an unstructured triangular 2D mesh and mesh extrusion to form a 3D mesh using prism elements if you want to avoid an unstructured tetrahedral mesh.
- When using identity-coupling variables to connect boundaries or edges that do not have equivalent meshes, use the sparse null-space function, particularly in 3D. Normally the automatic selection takes care of this. To specify the sparse null-space function, open the **Solver Parameters** dialog box, click the **Advanced** tab, and select **Sparse** in the **Null-space function** list.
- Use the same name for the dependent variables and the application modes that represent the same physics in different geometries. This makes it convenient to postprocess the results in the entire geometry. Select all the geometries in the **Geometries to use** list in the **Plot Parameters** dialog box to plot the results in all parts of the model. The default is that you have unique names, but it is possible to edit the names of the application mode and the dependent variable to be the same as long as the application modes belong to different geometries. For an example of how to do this, see “Example—Thin-Layer Diffusion” on page 442.

## IMPLEMENTATION OF THE IDENTITY CONDITIONS

COMSOL Multiphysics implements an identity condition as a special case of the extrusion coupling variable. In particular, it adds a variable name equal to the constraint name on the destination domains with a value equal to the constrained expression. The constraint works as an *ideal constraint*, that is, the derivatives of the constraint with respect to the solution help form the reaction forces. If you need more

freedom in specifying how the reaction forces are applied, use the extrusion coupling variable directly and specify the reaction force and constraint as weak contributions.

---

**Note:** It is normally much easier to work with an *assembly* that consists of two or more geometrical parts. You can then use identity pairs and interactive meshing to create a model where the physics is connected and where you can use different types of meshes in each part.

---

### MODELING PROCEDURE FOR MULTIGEOMETRY LINKING

The following list shows the main steps you must take to make a model with a continuous field that you link between two or more geometries:

- 1 Create the full geometry using separate geometries (parts) in the COMSOL Multiphysics model. Use the **View All 3D Geometries/View All 2D Geometries** button in the Visualization/Selection toolbar or the **View Geometries** dialog box to show other geometries than the current geometry during modeling. This helps to see that the geometries are spatially connected and that the total geometry looks right.
- 2 Create the meshes for the different geometries. These can be of different kinds, which is not possible using a single geometry. This is the main advantage with this multigeometry approach.
- 3 Add the physics to the different geometries. It is typically the same type of physics, and the field is continuous across the geometries. For this purpose, use the same name for the field variables (dependent variables) and the application modes. You can then visualize and postprocess the solution on the entire geometry using the same variable names for both dependent variables and application mode variables that use the application mode name as a suffix.

---

**Note:** The default names for the application modes and dependent variables are different for each application mode that you add. Make sure to edit these to make all names the same.

---

- 4 Use identity boundary conditions to link the physics across the different geometries.
- 5 Compute the solution.
- 6 For visualization plot the results in all geometries simultaneously. To do so, select all the geometries in the **Geometries to use** list in the **Plot Parameters** dialog box.

# Example—Thin-Layer Diffusion

The following example shows how to link two separate geometries into a 3D thin-layer diffusion model using identity boundary conditions. It also illustrates the use of different mesh element types; the brick and prism elements provide a large reduction in the number of degrees of freedom (DOFs) for this thin geometry. For more information about mesh elements and meshing options, see “Meshing” on page 299 of the *COMSOL Multiphysics User’s Guide*.

In this model you can significantly reduce the number of DOFs—and thus the solution time—by taking advantage of the geometry shape using brick and prism meshes instead of an unstructured tetrahedral mesh. The total model has only about 2300 DOFs using this approach. Using a default tetrahedral mesh on the same geometry leads to a problem with roughly 18,000 DOFs.

The physics in this model is a single-species diffusion. The dependent variable is the concentration,  $c$ . All boundaries are insulated except the inlet and the outlet. At the inlet boundary the concentration is  $c_0$ . At the outflow boundary (the bottom surface) there is an outward flux of  $-r_{\text{surf}}c/c_0$  (COMSOL Multiphysics defines the inward flux as positive), where  $r_{\text{surf}}$  is the reaction rate at the surface. An effective diffusion coefficient takes the porous material in the thin bottom plate into account. Table 15-2 on page 449 lists all material properties in the model.

The interface condition at the connection between the top and bottom parts is that the concentration  $c$  is equal on both sides.

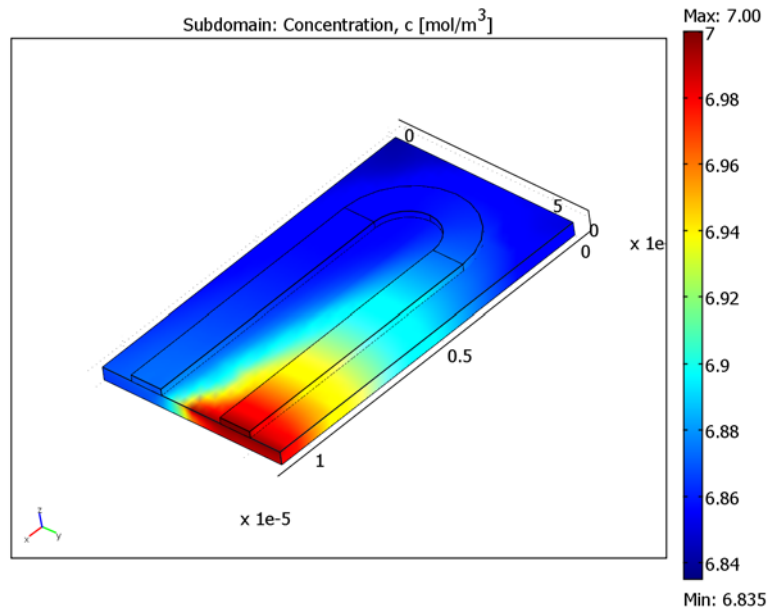


Figure 15-7: Resulting concentration distribution in the full geometry.

---

**Model Library path:** COMSOL\_Multiphysics/Diffusion/  
thin\_layer\_diffusion

---

### *Modeling Using the Graphical User Interface*

---

#### **MODEL NAVIGATOR**

- 1 Start COMSOL Multiphysics.
- 2 In the **Model Navigator** click the **Multiphysics** button.
- 3 Click the **Add Geometry** button to add a 2D geometry.
- 4 Click **OK** to accept the default settings and create the first geometry: **Geom1 (2D)**.
- 5 Repeat Steps 3 and 4 to create a second 2D geometry: **Geom2 (2D)**.
- 6 Select **Geom1 (2D)** to make it the current geometry.
- 7 Click **OK**.

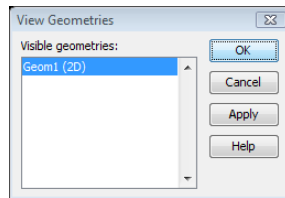
## GEOMETRY MODELING

### *Geometry 1 (Base Block Work Plane)*

- 1 Shift-click the **Rectangle/Square** button on the Draw toolbar.
- 2 In the **Rectangle** dialog box, type  $11e-6$  in the **Width** edit field and  $6e-6$  in the **Height** edit field.
- 3 Click **OK**.
- 4 Click the **Zoom Extents** button on the Main toolbar.

### *Geometry 2 (Top Layer Work Plane)*

- 1 Click the **Geom2** tab to switch to the other 2D geometry.
- 2 From the **Options** menu choose **View Geometries**.
- 3 In the **View Geometries** dialog box select **Geom1 (2D)**, then click **OK**.



You can also use the **View All 2D Geometries** button in the Visualization/Selection toolbar.

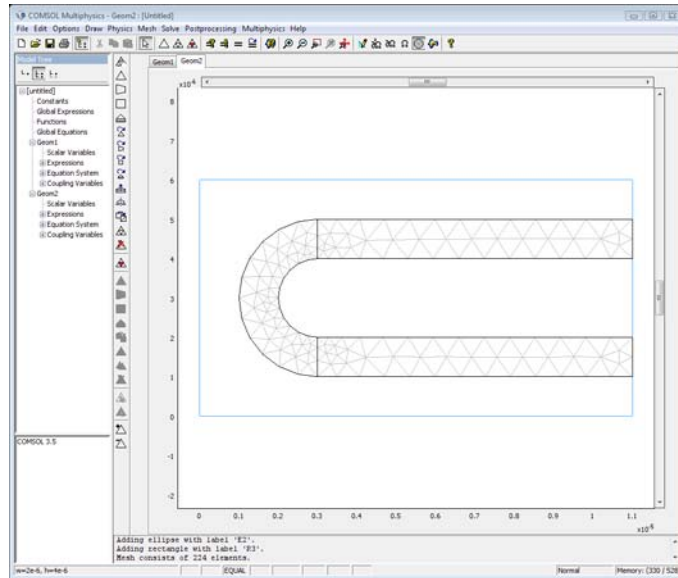
- 4 Click the **Zoom Extents** button on the Main toolbar.
- 5 Draw a rectangle with corners at  $(3e-6, 5e-6)$  and  $(1.1e-5, 4e-6)$ .
- 6 Press **Ctrl+C** to make a copy of the rectangle, then press **Ctrl+V** to paste the copy.
- 7 In the **Paste** dialog box, type  $-3e-6$  in the **y** edit field, then click **OK**.
- 8 Draw a centered circle, C1, with its center at  $(3e-6, 3e-6)$  and a radius of  $2e-6$  by first clicking the **Ellipse/Circle (Centered)** button on the Draw toolbar, and then using the right mouse button in the drawing area.
- 9 Draw another centered circle, C2, with its center at  $(3e-6, 3e-6)$  and a radius of  $1e-6$ .
- 10 Draw a rectangle, R3, with corners at  $(3e-6, 5e-6)$  and  $(5e-6, 1e-6)$ .
- 11 From the **Draw** menu choose **Create Composite Object**.
- 12 In the **Set formula** edit field, type  $C1 - C2 - R3$ . Click **OK**.



## MESH GENERATION

Geometry 2 (Top Layer Work Plane)

Click the **Initialize Mesh** button to create a triangular mesh for the top U-shaped part.

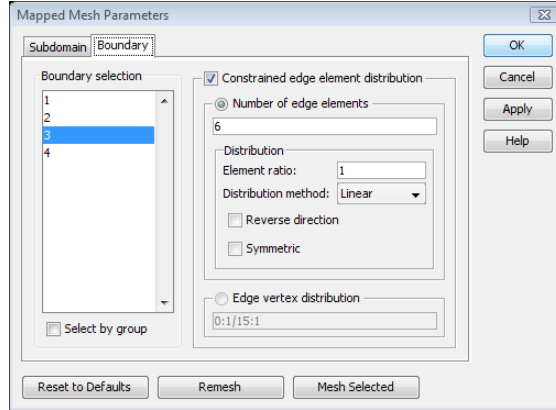


*The triangular mesh for the top layer.*

Geometry 1 (Base Block Work Plane)

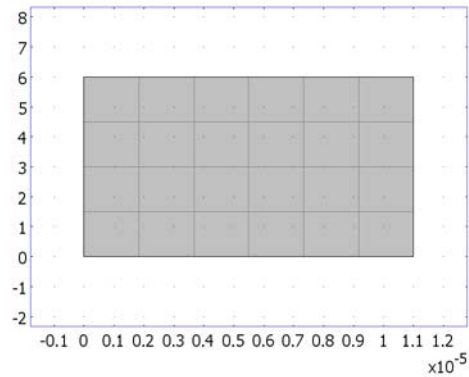
- 1 Click the **Geom1** tab.
- 2 From the **Mesh** menu, choose **Mapped Mesh Parameters**.
- 3 Click the **Boundary** tab.
- 4 Select Boundary 1.
- 5 Click to select the **Constrained edge element distribution** check box.
- 6 In the **Number of edge elements** edit field, type 4.
- 7 Select Boundary 3.
- 8 Click to select the **Constrained edge element distribution** check box.

9 In the **Number of edge elements** edit field, type 6.



10 Click the **Remesh** button. This creates a 4-by-6 quadrilateral mesh.

11 Click **Cancel**.



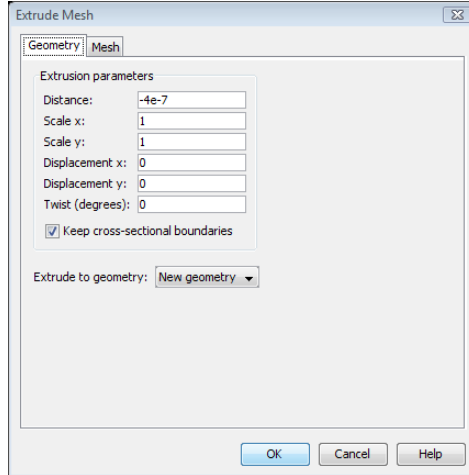
*The quadrilateral mesh for the base block work plane.*

Geometry 3 (Base Block)

The next step is to extrude the mapped mesh into a 3D brick mesh.

I From the **Mesh** menu, choose **Extrude Mesh**.

2 In the **Distance** edit field, type  $-4e-7$ .



3 Click **OK** to extrude the quadrilateral mesh 0.4 micrometer in the negative  $z$  direction. This creates a brick mesh in a new 3D geometry, **Geom3 (3D)**.

#### *Geometry 4 (Top Layer)*

1 Click the **Geom2** tab.

2 From the **Mesh** menu, choose **Extrude Mesh**.

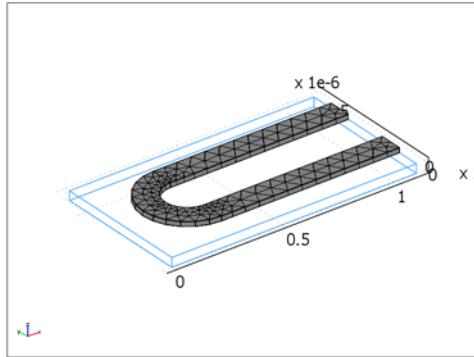
3 In the **Distance** edit field, type  $2e-7$ .

4 From the **Extrude to geometry** list, select **New geometry**.

5 Click **OK** to extrude the triangular mesh  $0.2 \mu\text{m}$  in the positive  $z$  direction. This creates a prism mesh in a new 3D geometry, **Geom4 (3D)**.

6 In the **View Geometries** dialog box, select **Geom3 (3D)**. Click **OK** to view both 3D geometries.

- 7 Click the **Zoom Extents** button on the Main toolbar.



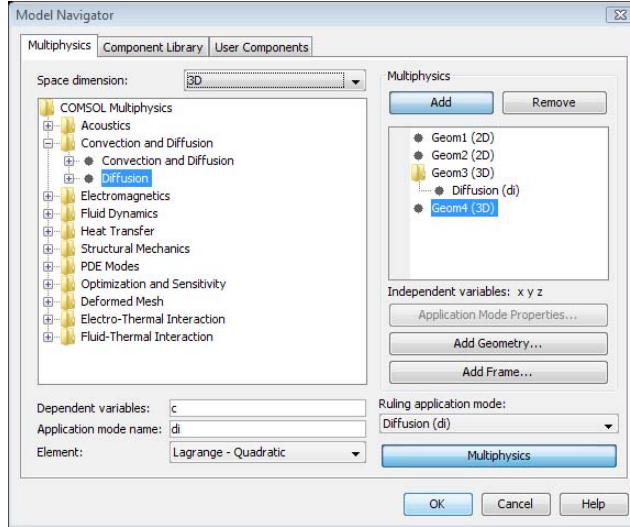
*The extruded prism mesh. The blue lines indicate the base block geometry.*

#### *Model Navigator*

Open the Model Navigator from the **Multiphysics** menu to add Diffusion application modes to both 3D geometries:

- 1 From the **Multiphysics** menu, choose the **Model Navigator**.
- 2 Select **Geom3 (3D)** in the list of geometries in the **Multiphysics** area.
- 3 In the list of application modes, open the **COMSOL Multiphysics > Convection and Diffusion** folder. Select **Diffusion**, then click **Add**.
- 4 Select **Geom4 (3D)** in the list of geometries in the **Multiphysics** area.  
Add another Diffusion application mode with the same name and dependent variable:
- 5 Make sure that **Diffusion** is selected in the list of application modes.
- 6 Type  $c$  in the **Dependent variables** edit field.

7 In the **Application mode name** edit field, type di.



8 Click **Add**, then click **OK**.

## OPTIONS AND SETTINGS

- 1 From the **Options** menu, choose **Constants**.
- 2 Enter the constants from the following table (the descriptions are optional); when finished, click **OK**.

TABLE 15-2: MODEL DATA

NAME	EXPRESSION	DESCRIPTION
r_surf	0.005[ $\text{mol}/(\text{m}^2\cdot\text{s})$ ]	Reaction rate, outlet
c0	7[ $\text{mol}/\text{m}^3$ ]	Concentration, inlet
D1	5e-5[ $\text{m}^2/\text{s}$ ]	Diffusion coefficient, top layer
D1_eff	1e-6[ $\text{m}^2/\text{s}$ ]	Effective diffusion coefficient, base block

## PHYSICS SETTINGS

### *Subdomain Settings—Base Block*

- 1 Click the **Geom3** tab.
- 2 In the **View Geometries** dialog box, select **Geom4 (3D)**. Click **OK** to view both 3D geometries.
- 3 From the **Physics** menu, choose **Subdomain Settings**.

- 4 Select Subdomain 1.
- 5 Type  $D1\_eff$  in the **D (isotropic)** edit field for the isotropic diffusion coefficient.
- 6 Click the **Init** tab.
- 7 In the  $c(t_0)$  edit field, type  $c0$ .
- 8 Click **OK**.

#### *Boundary Conditions—Base Block*

- 1 From the **Physics** menu, choose **Boundary Settings**.
- 2 Select Boundary 3 (the bottom boundary).
- 3 From the **Boundary condition** list, select **Flux**.
- 4 Type  $-r\_surf*c/c0$  in the  $N_0$  edit field for the inward flux to define an outward flux.
- 5 Click **OK**.

#### *Subdomain Settings—Top Layer*

- 1 Click the **Geom4** tab.
- 2 In the **View Geometries** dialog box, select **Geom3 (3D)**. Click **OK** to view both 3D geometries.
- 3 From the **Physics** menu, choose **Subdomain Settings**.
- 4 Press Ctrl+A to select all subdomains.
- 5 Type  $D1$  in the **D (isotropic)** edit field for the isotropic diffusion coefficient.
- 6 Click the **Init** tab.
- 7 In the  $c(t_0)$  edit field, type  $c0$ .
- 8 Click **OK**.

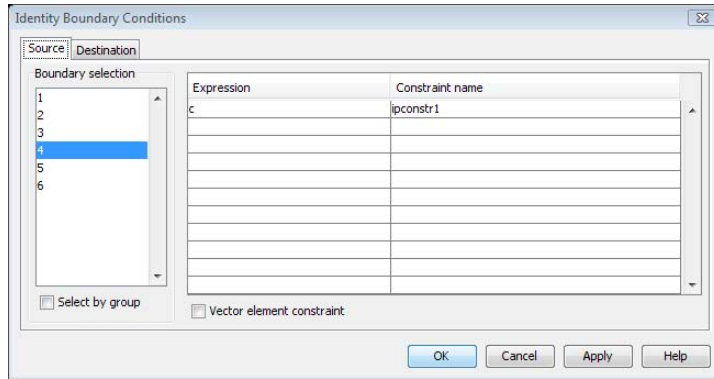
#### *Boundary Conditions—Top Layer*

- 1 From the **Physics** menu, choose **Boundary Settings**.
- 2 Select Boundary 18 (the top inflow boundary).
- 3 Select **Concentration** in the **Boundary condition** list.
- 4 In the  $c_0$  edit field for concentration, type  $c0$ .
- 5 Click **OK**.

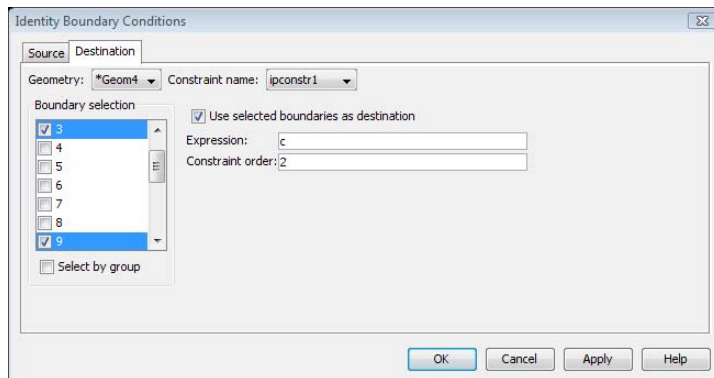
#### *Identity Boundary Conditions*

To couple the concentration values from the base block to the top layer, use an identity boundary condition.

- 1 Click the **Geom3** tab.
- 2 In the **Options** menu, point to **Identity Conditions**, and then click **Identity Boundary Conditions**.
- 3 Select Boundary 4 (the top boundary).
- 4 Type **c** in the top row under **Expression**.
- 5 Move to the **Constraint name** column. The default name **ipconstr1** appears automatically.



- 6 Click the **Destination** tab.
- 7 From the **Geometry** list, select **Geom4**.
- 8 Select Boundaries 3, 9, and 14 in the user interface. Click to select the **Use selected boundaries as destination** check box or click to select these boundaries directly in the **Boundary selection** list.
- 9 In the **Expression** edit field, type **c**.



- 10 Click **OK**.

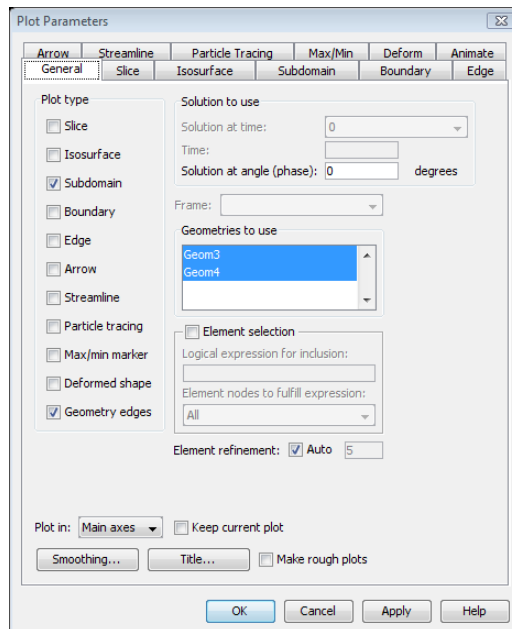
### Computing the Solution

Click the **Solve** button on the Main toolbar to start the analysis.

### Postprocessing and Visualization

To show the concentration distribution in the entire geometry, do the following steps:

- 1 From the **Postprocessing** menu, choose **Plot Parameters**.
- 2 On the **General** page, select both **Geom3** and **Geom4** in the **Geometries to use** list.
- 3 In the **Plot type** area click to clear the **Slice** check box, then select the **Subdomain** check box.



- 4 Click **OK**.



## Deformed Meshes

This chapter explains how to use the application modes that control mesh deformation: the Moving Mesh (ALE) application mode and the Parameterized Geometry application mode.

# Deformed Mesh Fundamentals

A deformed mesh can be useful if the boundaries of your computational domain are moving in time or as a function of a parameter. The point is that a new mesh need not be generated for each configuration of the boundaries—instead, the software simply perturbs the mesh nodes so they conform with the moved boundaries.

In COMSOL Multiphysics, you can control the movement of the interior nodes in three ways:

- By propagating the moving boundary displacement throughout the domain to obtain a smooth mesh deformation everywhere — this is done by solving PDEs for the mesh displacements (a Laplace or Winslow smoothing PDE) with boundary conditions given by the movement of the boundaries.
- By specifying an explicit formula for the mesh deformation.
- By letting the mesh movement be determined by some physical deformation variables, such as the displacement components of structural mechanics.

The technique for mesh movement is called an *arbitrary Lagrangian-Eulerian* (ALE) method. In the special case of a *Lagrangian* method, the mesh movement follows the movement of the physical material. Such a method is often used in solid mechanics, where the displacements often are relatively small.

When the material motion is more complicated, like in a fluid flow model, the Lagrangian method is not appropriate. For such models, an *Eulerian* method, where the mesh is fixed, is often used—except that this method cannot account for moving boundaries.

The ALE method is an intermediate between the Lagrangian and Eulerian methods, and it combines the best features of both—it allows moving boundaries without the need for the mesh movement to follow the material.

## *Mathematical Description of the Mesh Movement*

---

Though moving meshes are also possible in 3D, consider a 2D geometry for simplicity. First denote the spatial coordinates as  $(x, y)$  and then let  $(X, Y)$  be the coordinates of a mesh node in the initial, undeformed configuration. To describe the spatial coordinates  $(x, y)$  of the same mesh node in the deformed configuration, you can use the functions

$$x = x(X, Y, t), \quad y = y(X, Y, t)$$

where  $t$  is time or some other parameter. This coordinate transformation relates two *frames* (coordinate systems):

- The *spatial frame* is the usual, fixed coordinate system with the *spatial coordinates*  $(x,y)$ . In this coordinate system the mesh is moving, that is, the coordinates  $(x,y)$  of a mesh node are functions of time.
- The *reference frame* is the coordinate system defined by the *reference coordinates*  $(X,Y)$ . In this coordinate system the mesh is fixed to its initial position. You can view the reference frame as a curvilinear coordinate system that follows the mesh.

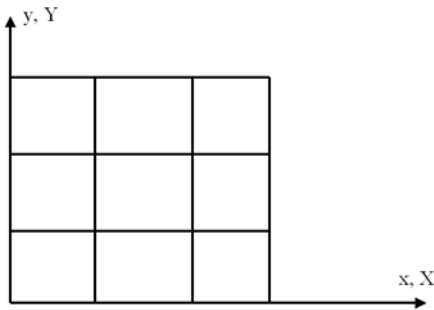


Figure 16-1: An undeformed mesh. The spatial frame  $(x, y)$  and the reference frame  $(X, Y)$  coincide.

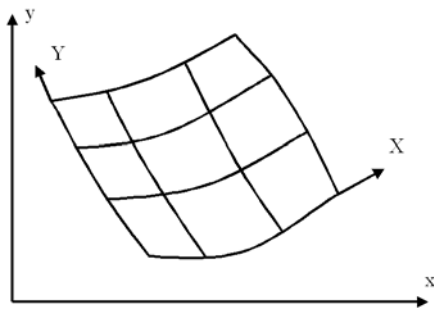


Figure 16-2: A deformed mesh with spatial frame  $(x, y)$  and reference frame  $(X, Y)$ .

Some geometric variables that the software defines are available for both the spatial and the reference frames (see “Geometric Variables” on page 170 in the *COMSOL Multiphysics User’s Guide*).

### SPATIAL DERIVATIVES

For a dependent variable  $u$ , there are two kinds of spatial derivatives:

- $u_x$  and  $u_y$ , the derivatives in the spatial frame. That is,  $u$  is considered as a function of  $x$  and  $y$  (and  $t$ ). These are typically denoted  $u_x$  and  $u_y$  in the software.
- $u_X$  and  $u_Y$ , the derivatives in the reference frame. That is,  $u$  is considered as a function of  $X$  and  $Y$  (and  $t$ ). These are typically denoted  $u_X$  and  $u_Y$  in the software.

In COMSOL Multiphysics it is only possible to use one of these types of derivatives depending on the frame associated with the application mode that defines  $u$ .

### TIME DERIVATIVES

There are two kinds of time derivatives related to the two frames:

- The common *spatial time derivative* is related to the spatial frame:

$$u_t = \left. \frac{\partial u}{\partial t} \right|_{x,y}$$

The spatial coordinates  $(x, y)$  are fixed when computing this derivative; that is, you consider a fixed point in space. This derivative is often denoted  $u_t$  in the software (see page 461).

- The *reference time derivative* (or *mesh time derivative*) is related to the reference frame:

$$u_{\text{TIME}} = \left. \frac{\partial u}{\partial t} \right|_{X,Y}$$

The reference coordinates  $(X, Y)$  are fixed when computing this derivative; that is, you consider a point that follows the moving mesh, for instance a mesh node. This derivative is denoted  $u_{\text{TIME}}$  in the software.

The two derivatives are related by the chain rule:

$$u_t = u_{\text{TIME}} - u_x x_{\text{TIME}} - u_y y_{\text{TIME}}$$

where  $(x_{\text{TIME}}, y_{\text{TIME}})$  is the mesh velocity. The reference time derivative is often less important from the user point view because its value depends on the mesh movement, which has no physical significance. However, for the special case when the mesh

follows the material's motion (the Lagrangian method), the reference time derivative is physically significant and is also called the *material time derivative*.

The solvers assemble the discretized model on the deformed mesh, using the reference time derivative of the solution as input. They also compute the spatial time derivative by the preceding formula. This implies that the PDEs do not have to be reformulated when you use a deformed mesh. However, if you use a Lagrangian method, some material properties might depend on the material deformation (which is then the same as the mesh deformation).

# Frames for Deformed Meshes

When modeling deformed meshes you need two *frames*. A *frame* is a coordinate system, and it includes a set of coordinate names. The two frames are:

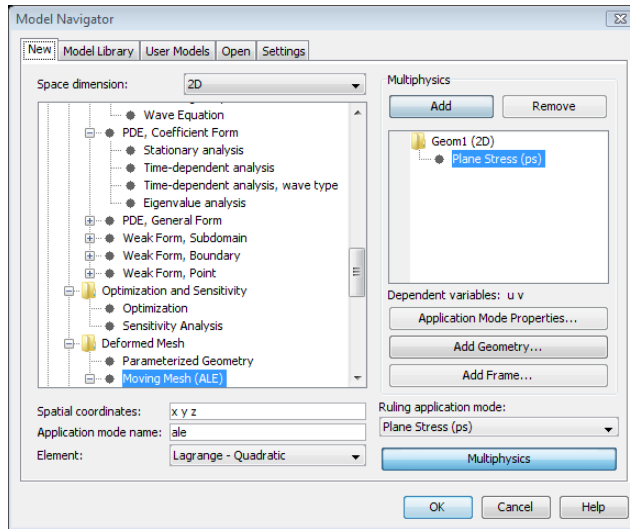
- The *reference frame*, which is the frame in which the mesh is fixed. This is the frame in which you draw the geometry and gives the shape of the geometry before it is deformed. This frame is associated with the geometry, and it exists even when you do not use deformed meshes.
- The *spatial frame*, which is the usual coordinate system that is fixed in space. In this frame the mesh is moving according to the settings in the deformed mesh application mode. The spatial frame is associated with the deformed mesh application mode.

## *Frames in the Model Navigator*

---

Before you have added a deformed mesh application mode, there is only one frame, the reference frame. The reference coordinates are the independent variables in the

geometry,  $x$ ,  $y$ ,  $z$ , by default. You can change these names and the name of the reference frame in the **Add Geometry** dialog box.

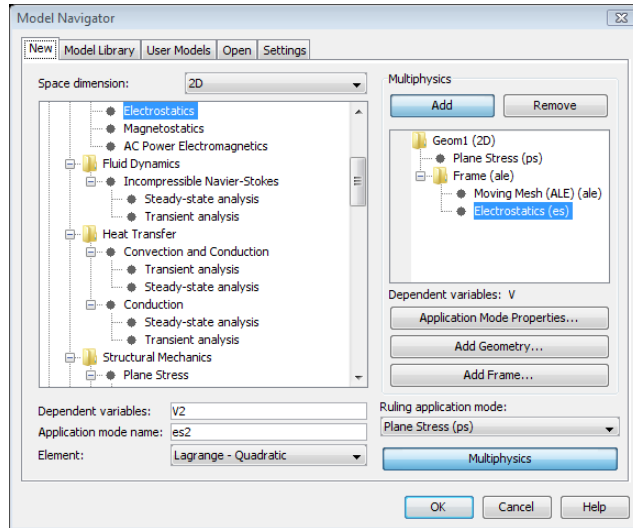


*Figure 16-3: The Model Navigator with a deformed mesh application mode selected in the list to the left.*

When you select a deformed mesh application mode from the list of application modes to the left in the Model Navigator, you can specify the names of the spatial coordinates in an edit field below the list. By default they take the names of the independent variables of the geometry. If you do not change the default, the independent variables of the geometry (the reference coordinates) get new names. The software creates the names by capitalizing the spatial coordinates. For example, if the spatial coordinates are  $x$ ,  $y$ , and  $z$ , it renames the reference coordinates to  $X$ ,  $Y$ , and  $Z$ . If you edit the default spatial coordinates so that they are not equal to the current reference coordinates, the reference coordinates retain their names.

When you add a deformed mesh application mode, the software automatically adds a spatial frame. Its name coincides with the name of the deformed mesh application mode with which it is associated (either `ale` or `pg`). Alternatively, you can add the spatial frame in advance by going to the **Multiphysics** area on the right side of the Model Navigator and clicking the **Add Frame** button. This opens the **Add Frame** dialog box, where you can specify the names of the spatial coordinates and the spatial frame. If you then add a deformed mesh application mode, it is associated with the previously defined frame.

The **Multiphysics** area to the right displays the spatial frame as a node attached to the geometry node. The deformed mesh application mode—in this case a Moving Mesh (ALE) application mode—is attached to the spatial frame it defines. All other application modes are attached to the frame in which their equations are formulated. Application modes attached directly to the geometry are associated with the reference frame.

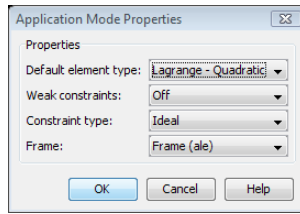


*Figure 16-4: The Model Navigator with a Moving Mesh (ALE) application mode in the model. The Plane Stress application mode's equations are formulated on the reference frame and the Electrostatics application mode's equations are formulated on the frame defined by the Moving Mesh (ALE) application mode.*

When you add an application mode, it becomes attached to the reference frame or the spatial frame depending on which node you selected in the **Multiphysics** area. If you



want to change the frame to which an application mode is attached, open the Application Mode Properties dialog box and select a frame in the **Frame** list



*Figure 16-5: The Application Mode Properties dialog box. The Frame property indicates which frame the application mode uses.*

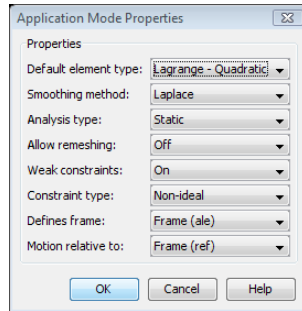
In normal situations, follow these guidelines when attaching an application mode to a frame:

- When using the Parameterized Geometry application mode, attach all application modes to the spatial frame.
- When using the Moving Mesh (ALE) application mode, attach structural mechanics application modes to the reference frame. This is because a structural analysis computes the mesh displacements that you typically use as input for the Moving Mesh (ALE) application mode when it defines the spatial frame. Attach all other application modes to the spatial frame defined by the Moving Mesh (ALE) application mode. In, for example, fluid-structure interaction models, consider taking the area effect into account, because the fluid force is defined on the deformed mesh in the spatial frame whereas the structural mechanics loads use the fixed reference frame. You specify this factor as the mesh element scale factor for the spatial frame divided by the mesh element scale factor for the reference frame. This expression is typically  $dv01\_ale/dv01$ . You find the names of the mesh element scale factors in the **Frames** area of the **Model Settings** dialog box on the **Physics** menu.

An application mode's frame affects the names of its variables. Specifically, the software uses the frame's coordinate names when forming derivatives and vector components. In addition, it interprets a time derivative  $ut$  as a derivative in the application mode's frame, while  $uTIME$  is always a derivative in the reference frame. See also "Shape Function Variables" on page 177 of the *COMSOL Multiphysics User's Guide*.

The two application modes defining the deformed mesh have two frame properties:

- The **Defines frame** property indicates which frame’s spatial coordinates that the application mode defines (the spatial frame). This frame cannot be the reference frame associated with the geometry.
- The **Motion relative to** property indicates which frame serves as the reference frame. By default this is the reference frame associated with the geometry. It cannot be the same frame as the frame that the application mode defines.



*Figure 16-6: The Application Mode Properties dialog box of the Moving Mesh (ALE) application mode.*

### *Settings for Frames*

---

In the **Model Settings** dialog box, which you open from the **Physics** menu, you can control some settings for each frame (see “Model Settings with Multiple Frames” on page 187 of the *COMSOL Multiphysics User’s Guide*). In particular, you can select the order of the shape functions that describe the moving mesh. This order must agree with the element order you choose in the **Subdomain Settings** dialog box for the deformed mesh application mode (see “Subdomain Settings” on page 463). The default choice **Automatic** takes care of this.

### *Frame Selection for Coupling Variables*

---

If you work with coupling variables, periodic conditions, or identity conditions when you have multiple frames, you need to select the appropriate frame in the corresponding dialog boxes. See “Integrating on a Deformed Mesh” on page 271 of the *COMSOL Multiphysics User’s Guide* for details.

# The Moving Mesh Application Mode

With the Moving Mesh (ALE) application mode you can create models where the geometry (actually the mesh) changes shape due to some physics in the model. You can study both static and time-dependent deformations where the geometry changes its shape due to the dynamics of the problem.

## *Subdomain Settings*

The **Subdomain Settings** dialog box in this application mode contains properties pertaining to the moving mesh.

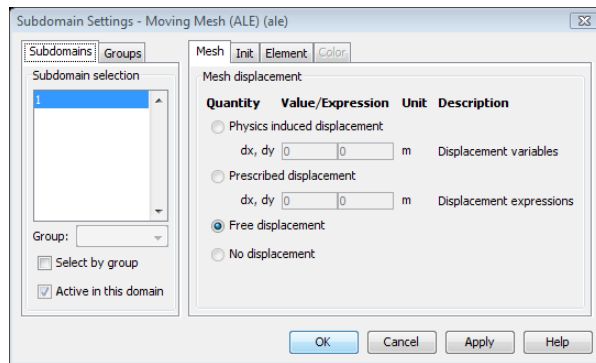


Figure 16-7: The Subdomain Settings dialog box of the Moving Mesh application mode.

On the **Mesh** tab you can specify how the software computes the mesh displacement in each subdomain. The available options are:

- **Free displacement.** This is the default setting, and it means that the mesh displacement is constrained only by the boundary conditions on the surrounding boundaries. The displacement in the subdomain is obtained by solving a PDE (see “Smoothing Methods” on page 466).
- **Physics induced displacement.** Use this option when another application mode calculates the displacement. Normally that other application mode is a structural mechanics application mode, and the displacement variables to enter in the edit fields are the dependent variables of the structural mechanics application mode. In other words, a Lagrangian method is used where the mesh movement follows the material motion. The contents of the edit fields for the displacement variables must be a single dependent variable discretized with Lagrange shape functions. The order

of these shape functions must be compatible with the order chosen on the **Element** tab.

- **Prescribed displacement.** Use this when expressions define the displacement. These should not depend on variables that the model solves. If they do, this dependence does not contribute to the Jacobian matrix, which can cause convergence difficulties. This is useful, for example, in axisymmetric structural models, where the dependent variable represents  $u/r$  rather than the displacement  $u$  itself, which you have access to as an expression variable.
- **No displacement.** The mesh is not deformed at all.

The physics-induced displacement and prescribed displacement might seem very similar because they both require that you enter the displacements as expressions. The difference is that you can use the physics-induced displacement only when the displacement is equal to the dependent variables of an application mode. You cannot use it for general expressions. The prescribed displacement, on the other hand, assumes that the expressions are functions of variables whose values are known (the reference coordinates, for instance).

On the **Init** tab you can give initial values for the mesh positions  $x$ ,  $y$ , and  $z$  (if the Laplace smoothing method is used), or the mesh displacements  $dx$ ,  $dy$ , and  $dz$  (if the Winslow smoothing method is used).

On the **Element** tab you can select the order of the Lagrange shape functions that define the mesh positions  $x$ ,  $y$ , and  $z$  (if the Laplace smoothing method is used), or the mesh displacements  $dx$ ,  $dy$ , and  $dz$  (if the Winslow smoothing method is used). The shape functions are only used on subdomains where **Free displacement** has been selected. You can also select the **Integration order** and **Constraint order** used when discretizing the PDE for the mesh movement and its boundary conditions.

## Boundary Conditions

At the boundaries you can set constraints on the mesh displacement or, in the transient case, on the mesh velocity. You can set constraints on the exterior boundaries of the subdomains where you use free displacement.

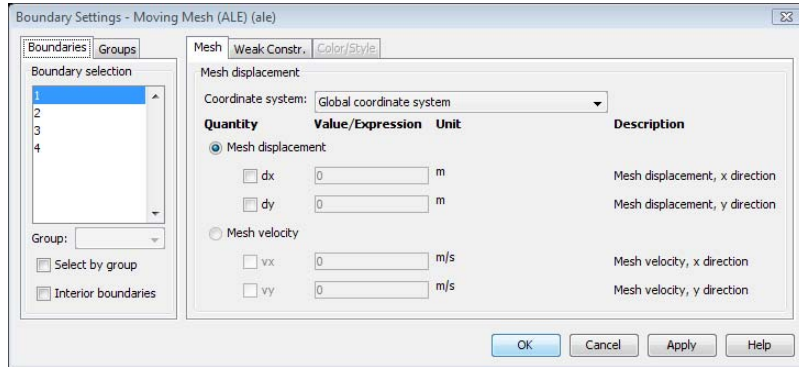


Figure 16-8: Boundary Settings dialog box of the Moving Mesh application mode.

If you want to constrain a certain displacement component, select its check box and enter an expression in the **Mesh displacement** edit field. If the check box is not selected, the boundary gets a Neumann boundary condition. You can also select in which coordinate system the displacement is specified.

For example, if you have a free boundary, it is appropriate to constrain the normal component of the mesh velocity to be equal to the normal component of the material velocity. You can do this by selecting the **Tangent and normal** coordinate system, entering the expression  $n_x * u + n_y * v$  for the normal velocity, and using a Neumann condition on the tangential velocity. This setup allows the mesh to slip relative to the material along the boundary, which improves mesh quality. Note that  $(n_x, n_y)$  denotes the normal of the deformed mesh, while  $(n_X, n_Y)$  denotes the normal of the undeformed mesh. Similar naming conventions hold for other geometric variables, see “Geometric Variables” on page 170 of the *COMSOL Multiphysics User’s Guide*.

If the boundary condition involves variables from another application mode (like in the free boundary condition), the usual pointwise constraints or ideal weak constraints make unwanted modifications of the boundary condition for the other application mode. For this reason, the Moving Mesh (ALE) application mode uses non-ideal weak constraints by default.

It is important that the boundary conditions are consistent with the settings in the adjacent subdomains. If the subdomain settings define a certain displacement, the boundary must be displaced in the same way.

### *Smoothing Methods*

---

In the domains with free displacement, the Moving Mesh (ALE) application mode solves an equation for the mesh displacement. This equation smoothly deforms the mesh given the constraints you place on the boundaries. You can choose between *Laplace smoothing* and *Winslow smoothing*. To specify the smoothing methods, use the **Application Mode Properties** dialog box, which you access from the **Multiphysics** area in the Model Navigator or from the **Physics** menu.

To see how these smoothing methods differ, let  $x$  and  $y$  be the spatial coordinates of the frame which the application mode defines, and let  $X$  and  $Y$  be the reference coordinates of the reference frame. If you choose Laplace smoothing, the software introduces deformed mesh positions  $x$  and  $y$  as degrees of freedom in the model. In the static case, it solves the equation

$$\frac{\partial^2 x}{\partial X^2} + \frac{\partial^2 x}{\partial Y^2} = 0$$

and in the transient case it solves the equation

$$\frac{\partial^2 \partial x}{\partial X^2 \partial t} + \frac{\partial^2 \partial x}{\partial Y^2 \partial t} = 0$$

Similar equations hold for the  $y$  coordinate.

If you choose Winslow smoothing, the software solves the equation

$$\frac{\partial^2 X}{\partial x^2} + \frac{\partial^2 X}{\partial y^2} = 0$$

and does the same for  $Y$ . Equivalently,  $X$  and  $Y$  satisfy Laplace equations as functions of the  $x$  and  $y$  coordinates.

### *Remeshing*

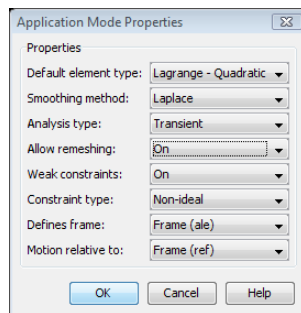
---

If the mesh displacement becomes large, the mesh elements eventually have a very bad quality or even become inverted. Depending on the specific model, either smoothing

technique may do a better job avoiding inverted mesh elements. Once a mesh element becomes inverted, it is no longer possible to solve any other equation on that frame. This section describes how to remesh the geometry when this happens and how to continue solving.

## ENABLING REMESHING

You enable the remeshing support in the Moving Mesh (ALE) application mode in the Application Mode Properties dialog box. Set the property **Allow remeshing** to **On**. You must make this selection before solving the model. If you attempt to solve it and later realize that you need to remesh while remeshing support is not activate, you must go back, activate it, and re-solve the model.



## FRAMES WHEN USING REMESHING

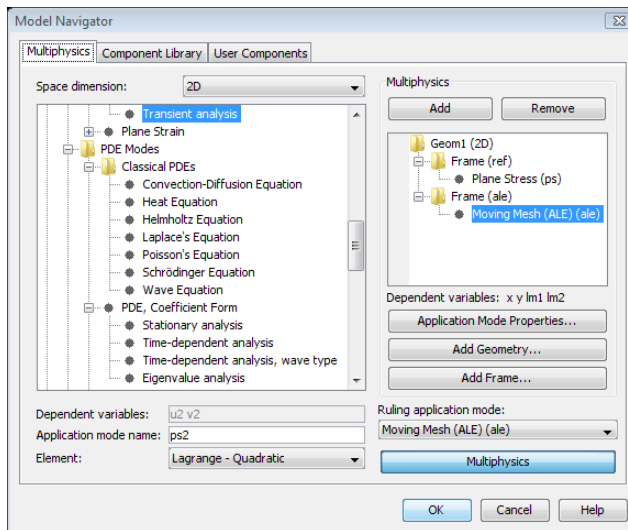
When you set the application mode property **Allow remeshing** to **On**, the software adds an extra frame. The three frames are now:

- The *spatial frame*, usually denoted by `a1e` or `pg` (the name of the application mode controlling the deformed mesh). This is the usual coordinate system. The corresponding coordinates are called `x` and `y` by default.
- The *reference frame*, usually denoted by `ref`. This coordinate system describes the original configuration (before remeshing). The corresponding coordinates are called `X` and `Y` by default. If the mesh movement follows the material movement (as is common in solid mechanics), this frame is also called the *material frame*. In this case, `X` and `Y` are the coordinates of the material point in the original configuration. The software stores the values of `X` and `Y` for all nodes as degrees of freedom in the

solution vector but never solves for them. Their values only change when mapping the solution, which happens when you click the **Restart** button after a remeshing.

- The *mesh frame*, usually denoted by **mesh**. This coordinate system describes the configuration just after the latest remeshing. The corresponding coordinates are called  $X_m$  and  $Y_m$  by default.

The guidelines on page 461 regarding which frame to attach the physics application modes to still apply. Note that no physics application mode should be attached to the mesh frame. In the **Model Navigator** the mesh frame is associated with the geometry and the reference frame is a separate node. This differs from the case without remeshing where the reference frame was associated with the geometry.



### STOP CONDITION

To make the solver stop when the mesh quality is poor you need to activate a stop condition in the **Solver Parameters** dialog box. Such a stop condition exists for the time-dependent solver and the parametric solver.

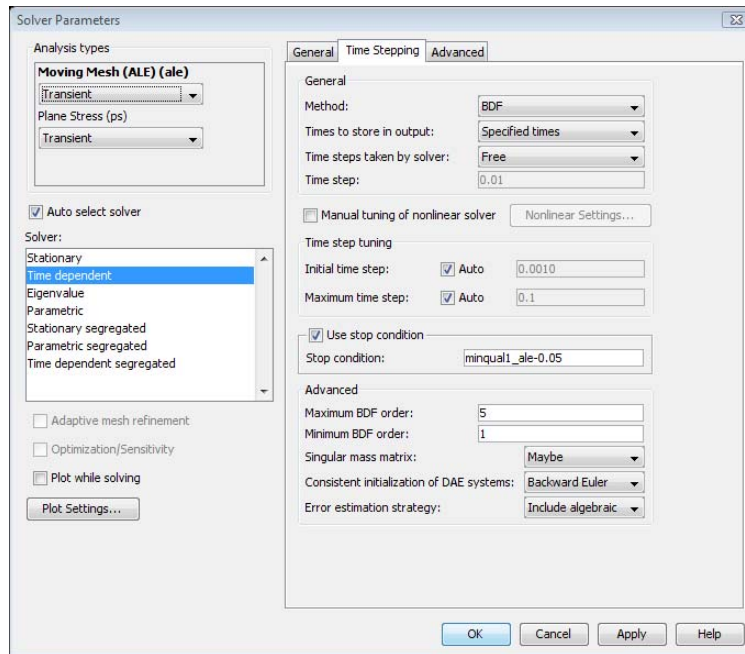
The solver halts when the expression in the **Stop condition** edit field becomes negative. The default value for the stop condition is  $\text{minqual1\_ale} - 0.05$ . The Moving Mesh (ALE) application mode defines the variable  $\text{minqual1\_ale}$ , which represents the minimum quality of the deformed mesh. The variable name begins with  $\text{minqual}$  followed by the geometry number. This nomenclature creates unique variables in the case where a model has multiple geometries. For models with one geometry this



number is 1, resulting in `minqual1`. The variable name for the minimum quality ends with an underscore (`_`) followed by the name of the frame for the deformed mesh. In this case, the name of that frame is `ale`, which means that the name of the quality variable becomes `minqual1_ale`. To find the name of the frame, open the **Model Navigator** and see to which frame the Moving Mesh (ALE) application mode is attached. You can also look in the **Application Mode Properties** dialog box. The **Defines frame** property gives the name of the frame of the deformed mesh.

The default stop condition makes the solver stop when the minimum quality is less than 0.05. Depending on the initial quality of the mesh, you might have to change this number.

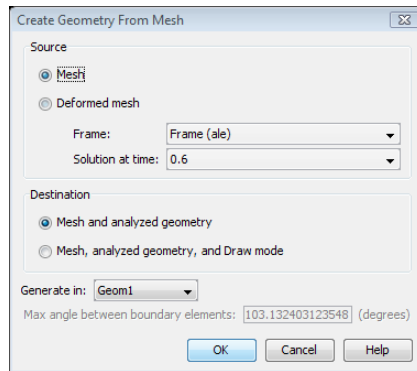
If the model has more than one geometry with a deformed mesh, the stop condition uses the minimum quality of all the geometries. In a model with two geometries, use the stop condition `min(minqual1_ale,minqual2_ale)-0.05`.



## REMESHING STEP BY STEP

To solve using remeshing of a deformed mesh follow these steps:

- 1 Activate the remeshing support in the application mode by setting the property **Allow Remeshing** to **On** in the **Application Mode Properties** dialog box.
- 2 Specify the stop condition in the **Solver Parameters** dialog box.
- 3 Solve. If the solver stops before it has reached the final time continue with the next step below.
- 4 Next you need to create a new geometry from the deformed mesh. From the **Mesh** menu, open the **Create Geometry From Mesh** dialog box. Click the **Deformed Mesh** option button to use the deformed mesh as the source for the new geometry. You can choose at which time to pick the mesh but in general use the last one.



- 5 Mesh the new geometry just created. This mesh gets a better quality than the previous deformed mesh.
- 6 In the **Solver Parameters** dialog box change the initial time (or the first parameter value if you are using the parametric solver) to the last value that the solver reached.
- 7 Click the **Restart** button to continue solving. This uses the previous solution as the initial value and thus continue solving where the solver had stopped.
- 8 If the solver again stops before the final time go to Step 4 again.

## REMESHING—A SIMPLE EXAMPLE

This section describes the remeshing technique by building a simple model where a force bends a beam.

- 1 In the **Model Navigator**, select **2D** from the **Space dimension** list.
- 2 In the **COMSOL Multiphysics** folder, select **Deformed Mesh>Moving Mesh (ALE)>Transient analysis**.

- 3 Click the **Multiphysics** button and then click the **Add** button to add the Moving Mesh application mode to the model.
- 4 In the list of application modes, select **Structural Mechanics>Plane Stress>Transient analysis**. In the **Multiphysics** area select the node **Geom1 (2D)**. Then click **Add** to add the Plane Stress application mode.
- 5 Click **OK** to close the **Model Navigator**.

#### *Geometry Modeling*

- 1 Draw a rectangle with a width of 1.6, height of 1.2, and with its lower left corner at  $(-0.6, -0.4)$ .
- 2 Draw a second rectangle with a width of 1.4, height of 0.2, and its lower left corner at  $(-0.6, -0.2)$ .

#### *Physics Settings*

- 1 In the **Subdomain Settings** dialog box of the Plane Stress application mode clear the **Active in this domain** check box for Subdomain 1 (the air).
- 2 You also need to add some damping to stabilize the transient solution. To do so, select Subdomain 2 (the beam), and then click the **Damping** tab.
- 3 Select **Rayleigh** from the **Damping** list.
- 4 Type 1 in the  $\alpha_{dM}$  edit field and 0.001 in the  $\beta_{dK}$  edit field.
- 5 Click **OK** to close the dialog box.
- 6 In the **Boundary Settings** dialog box select Boundary 3 and activate the constraints  $R_x$  and  $R_y$  by selecting the corresponding check boxes. Go to the **Load** page and select Boundaries 4, 6, and 8. Enter a force that increases with time by setting  $F_y$  to  $-1[\text{MN/m/s}] * t$ . Click **OK** to close the dialog box.
- 7 From the **Multiphysics** menu, select the **Moving Mesh (ALE) (ale)** application mode.
- 8 In the **Subdomain Settings** dialog box of the **Moving Mesh** application mode select Subdomain 2 and click the **Physics induced displacement** button. Enter the displacement variables  $u$  and  $v$  in the **dx** and **dy** edit fields. Click **OK** to close the dialog box.
- 9 In the **Boundary Settings** dialog box select Boundaries 4, 6, and 8. Select the **dx** and **dy** check boxes and then enter the mesh displacement  $u$  and  $v$  in the corresponding edit fields.
- 10 Select Boundaries 1, 2, 5, 7, and 9. Select the **dx** and **dy** check boxes and leave the mesh displacements at their default zero values to prevent these boundaries from moving. Click **OK** to close the dialog box.

---

**Note:** To obtain a correct solution, you should use the support for large deformations in the Plane Stress application mode, but that feature is available only in the extended Plane Stress application mode in the Structural Mechanics and MEMS modules.

---

The next step is to activate support for remeshing in the **Moving Mesh** application mode:

- 1 Open the **Application Mode Properties** dialog box by selecting **Properties** from the **Physics** menu.
- 2 Select **On** from the **Allow remeshing** list, then click **OK**.

You must make this selection before solving the model. If you attempt to solve it and later realize that you need to remesh while remeshing support is not activate, go back, activate it, and then re-solve the model.

#### *Mesh Generation*

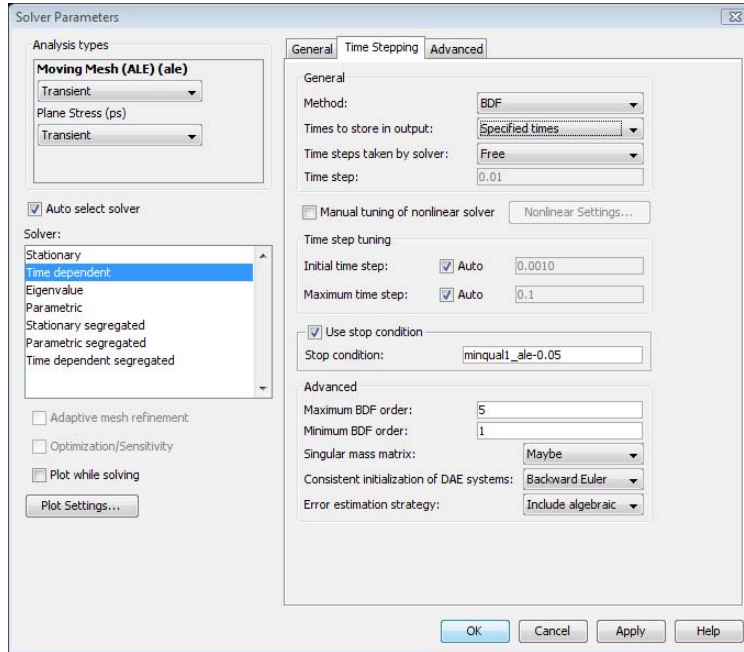
- 1 Initialize the mesh.
- 2 Refine it once.

#### *Computing the Solution*

To make the solver stop when the mesh quality becomes poor, first enable a stop condition in the **Solver Parameters** dialog box.

- 1 From the **Solve** menu open the **Solver Parameters** dialog box. In the **Times** edit field enter 0:0.01:0.3.

2 Go to the **Time Stepping** page and select the **Use stop condition** check box.



3 Click **OK** to close the **Solver Parameters** dialog box and then click the **Solve** button on the Main toolbar.

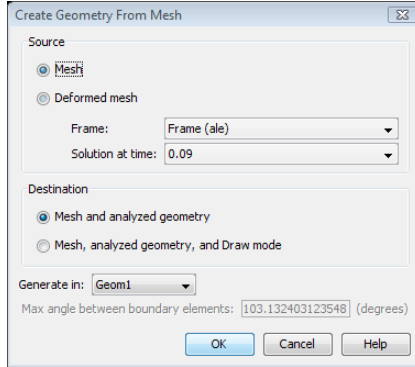
The solver stops somewhere around  $t = 0.12$ .

### *Geometry Modeling*

Next you need to create a new geometry and mesh from the deformed mesh.

1 From the **Mesh** menu open the **Create Geometry From Mesh** dialog box.

- 2 Click the **Deformed mesh** button, then click **OK**.



This creates a new geometry from the deformed mesh at the last time step.

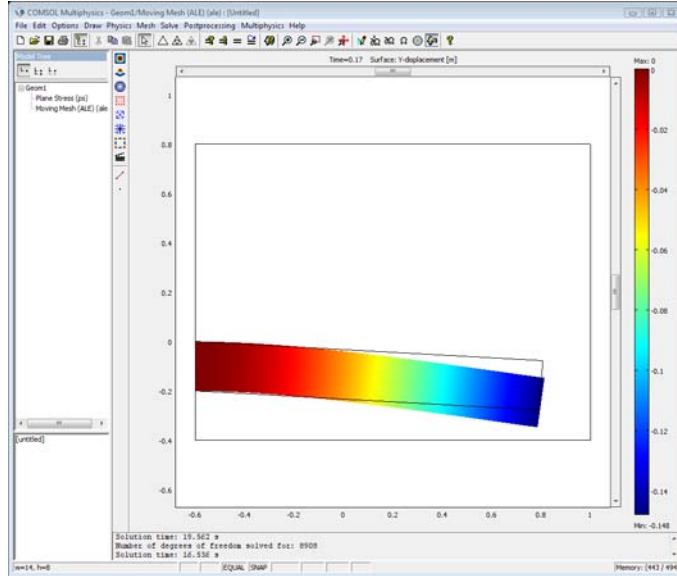
### *Mesh Generation*

Using buttons on the Main toolbar, initialize the mesh and refine it once. This creates a new mesh with higher quality for the deformed geometry.

### *Computing the Solution*

- 1 Open the **Solve>Solver Parameters** dialog box and on the **General** page change the list of times in the **Times** edit field to start from the last time of the current solution. Generally when solving time-dependent problems this is not absolutely necessary, but in this case it is important because the force in the Plane Stress application mode depends on time explicitly.
- 2 When you continue solving the model, it is important to use the solution from the last time step as the new initial value when restarting the solver. Do this by clicking the **Restart** button on the Main toolbar.

The solver continues solving and stops when the mesh quality again becomes poor.

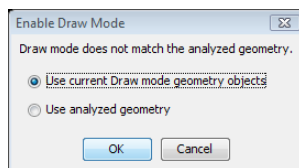


## POSTPROCESSING

To postprocess the two parts of the simulation together, you can export the FEM structure to the command line after each part of the simulation. Use **File>Export>Export FEM Structure** to save each FEM structure with a new name. Then use the ability of the `postmovie` command to postprocess several FEM structures.

## RESTORING THE ORIGINAL GEOMETRY

Assume you want to go back and restart the solver from the very beginning. In this case you first must restore the original geometry. To do so, go to the Draw mode. Because you have generated a deformed geometry, the objects in Draw mode do not represent the current geometry. The software therefore asks if you want to use the current objects in Draw mode or replace them with the deformed geometry (see “Entering Draw Mode” on page 36 in the *COMSOL Multiphysics User’s Guide*). Using the current Draw mode objects restores the original geometry.



# Example—Electrochemical Polishing

## *Introduction*

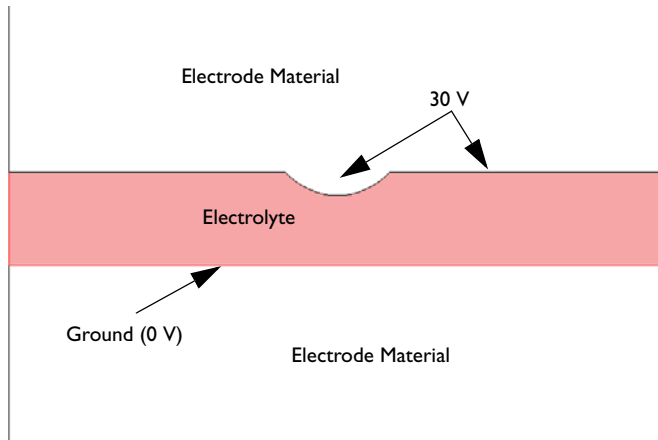
---

This example illustrates the principle of electrochemical polishing. The simplified 2D model geometry consists of two electrodes and an intermediate electrolyte domain. The positive electrode has a protrusion, representing a surface defect. The purpose of the model is to examine how this protrusion and the surrounding electrode material are depleted over a period of time.

## *Model Definition*

---

The potential drop over the electrodes is 30 V, and the electrolyte has a conductivity of 10 S/m.



Modeling the depletion of the positive electrode requires a moving boundary because the geometry changes and the current density distribution with it. A simple model for the depletion is based on the assumption that the depletion rate is proportional to the normal current density at the electrode surface. The velocity,  $U$ , normal to the mesh at the electrode surface then becomes

$$U = -KJ_n$$

where  $K$  is the coefficient of proportionality, and  $J_n$  is the normal current density. In this model,  $K = 10^{-11} \text{ m}^3/\text{As}$ .



The part of the electrode and electrolyte that the model includes is about 3 mm wide and the distance between the electrodes is 0.4 mm.

### *Modeling in COMSOL Multiphysics*

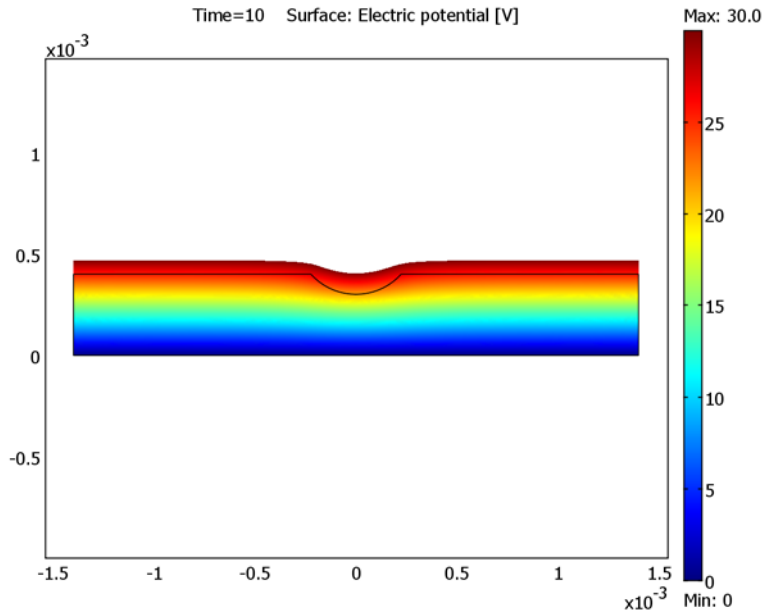
---

This model uses the Conductive Media DC and transient Moving Mesh (ALE) application modes. The variable for the normal current density defines the mesh velocity. The dynamics of the model is quasi-static in nature, and the time dependence only enters in the depletion (removal of material) of the electrode.

### *Results*

---

After 10 s, the protrusion is somewhat smoothed out, and a significant portion of the positive electrode has been depleted:



---

**Model Library path:** COMSOL\_Multiphysics/Electromagnetics/  
electrochemical\_polishing

---

### MODEL NAVIGATOR

- 1 Select **2D** in the **Space dimension** list. (This model can easily be generalized to 3D.)
- 2 In the application mode list, open **COMSOL Multiphysics>Deformed Mesh** and then **Moving Mesh (ALE)>Transient analysis**.
- 3 Click the **Multiphysics** button and then the **Add** button.
- 4 In the application mode list, open **COMSOL Multiphysics>Electromagnetics>Conductive Media DC**.
- 5 Click the **Add** button. This adds the **Conductive Media DC** application mode to the moving mesh frame.
- 6 Click **OK** to close the **Model Navigator**.

### OPTIONS

- 1 From the **Options** menu, open the **Constants** dialog box.
- 2 Enter a constant with the name **K**, the expression  $1e-11 [m^3 / (A*s)]$ , and the description **Coefficient of proportionality** (the description is optional).
- 3 Click **OK** to close the **Constants** dialog box.

### GEOMETRY MODELING

- 1 Click the **Rectangle/Square** button on the Draw toolbar, and draw a rectangle with corners at  $x = -1.4, y = 0$  and  $x = +1.4, y = 0.4$ .
- 2 Shift-click the **Ellipse/Circle (Centered)** button.
- 3 In the **Circle** dialog box, type **0.3** in the **Radius** edit field and set the center coordinates to **(0, 0.6)** by typing **0** in the **X** edit field and **0.6** in the **Y** edit field.
- 4 Click **OK**.
- 5 Select both objects by using the keyboard shortcut **Ctrl+A**. Both objects are now highlighted in red.
- 6 Click the **Difference** button on the Draw toolbar to remove the circle object from the rectangle. This creates the protruding part of the electrode.
- 7 The model needs to be the order of mm and not m. Click to select the object and then click the **Scale** button on the Draw toolbar.
- 8 Enter a scale factor of  $10^{-3}$  by typing **1e-3** in both the **X** and **Y** edit fields in the **Scale factor** area. Click **OK**.
- 9 Click the **Zoom Extents** button on the Main toolbar.

The geometry is now ready.

## PHYSICS SETTINGS

### *Subdomain Settings—Conductive Media DC*

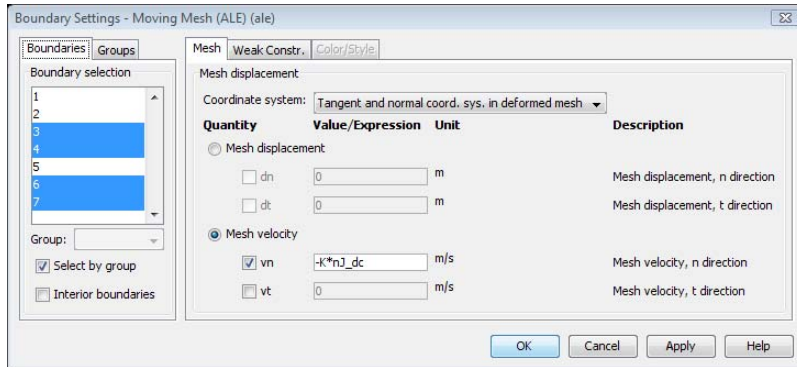
- 1 From the **Multiphysics** menu, select the **Conductive Media DC** application mode.
- 2 From the **Physics** menu, select **Subdomain Settings**.
- 3 In the **Subdomain Settings** dialog box, select Subdomain 1 (the only one) and type 10 in the **Electric conductivity** edit field.
- 4 Click **OK** to close the **Subdomain Settings** dialog box.

### *Boundary Settings—Conductive Media DC*

- 1 From the **Physics** menu, select **Boundary Settings**.
- 2 For Boundaries 3, 4, 6, and 7, select the **Electric potential** in the **Boundary condition** list and type 30 in the **Electric potential** edit field.
- 3 For Boundaries 1 and 5, select **Electric insulation** in the **Boundary condition** list. Electric insulation is a good approximation if you want to simulate that the electrodes are extended indefinitely in both directions.
- 4 For Boundary 2, select **Ground** as the boundary condition.
- 5 Click **OK** to close the **Boundary Settings** dialog box.

### *Boundary Settings—Moving Mesh (ALE)*

- 1 From the **Multiphysics** menu, select the **Moving Mesh (ALE)** application mode.
- 2 From the **Physics** menu, select **Boundary Settings**.
- 3 In the **Boundary Settings** dialog box, for boundary 2, click the **Mesh displacement** button and set the mesh displacement in both directions to 0 by selecting the **dx** and **dy** check boxes.
- 4 For Boundaries 1 and 5, click the **Mesh velocity** button and set the mesh velocity, in the  $x$  direction to 0 by selecting the **vx** check box.
- 5 Select Boundaries 3, 4, 6, and 7, and select **Tangent and normal coord. sys. in deformed mesh** in the **Coordinate system** list. Click the **Mesh velocity** button. Then select the **vn** check box and type  $-K \cdot nJ\_dc$  in the **Mesh velocity, n direction** edit field. This makes the normal mesh velocity equal to  $-KJ_n$  ( $nJ\_dc$  is the variable for the current density outflow from the Conductive Media DC application mode).
- 6 Click **OK**.



## MESH GENERATION

- 1 Click the **Initialize Mesh** button on the Main toolbar.
- 2 Click the **Refine Mesh** button on the Main toolbar once.

## COMPUTING THE SOLUTION

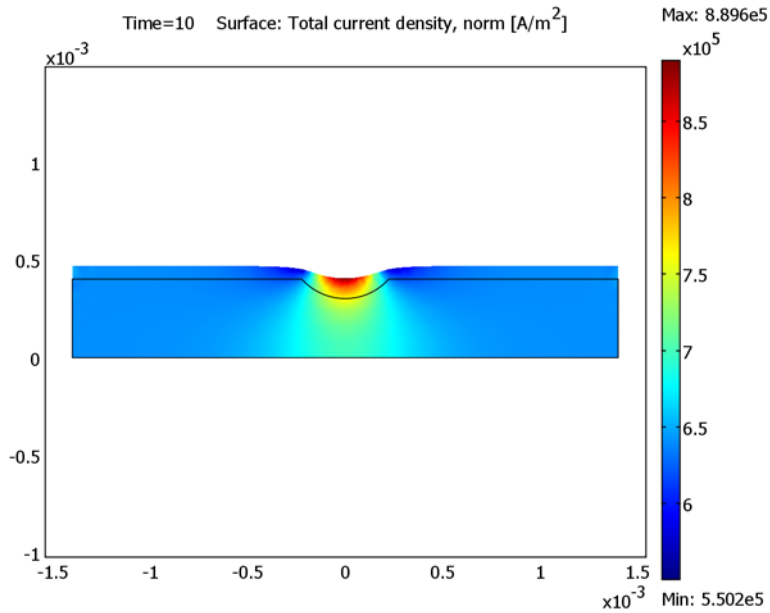
- 1 From the **Solve** menu, select **Solver Parameters**.
- 2 In the **Time stepping** area, type 0:1:10 for **Times**. This defines a simulation that runs from 0 to 10 s in steps of 1 s.
- 3 Click **OK** to close the **Solver Parameters** dialog box.
- 4 Click the **Solve** button (equal sign) on the Main toolbar to solve the model (or choose **Solve Problem** from the **Solve** menu).

## POSTPROCESSING AND VISUALIZATION

The default plot shows the potential distribution on the deformed mesh. Change the visualization settings to plot the current density distribution:

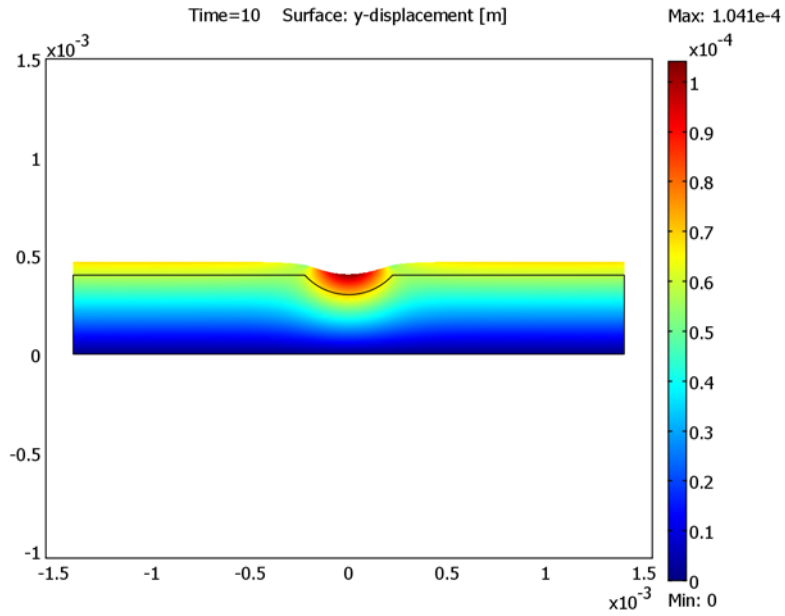
- 1 From the **Postprocessing** menu, select **Plot Parameters**.
- 2 On the **Surface** page, select **Conductive Media DC (dc)>Total current density, norm** from the **Predefined quantities** list.

3 Click **OK** to plot.



The maximum current density appears to be of the order of  $10^6$  A/m<sup>2</sup>. To see the magnitude of the depletion in the  $y$  direction more easily, plot the ALE-displacement variable  $dy$ :

- 1 From the **Postprocessing** menu, select **Plot Parameters**.
- 2 On the **Surface** page, type `dy_a1e` in the **Expression** edit field.
- 3 Click **OK** to plot.



The maximum value for the  $y$ -displacement is approximately  $10^{-4}$  m (or 0.1 mm).

You can now compare this with an approximate formula for the total depletion increment,  $d$ :

$$d = |U|\Delta t = K|J_n|\Delta t = \left(10^{-11} \frac{\text{m}^3}{\text{As}}\right) \cdot \left(10^6 \frac{\text{A}}{\text{m}}\right) \cdot (10^1 \text{s}) = 10^{-4} \text{ m}$$

This shows that the approximate formula (which does not take effects from the curved boundary into account) is in fact very accurate.

# The Parameterized Geometry Application Mode

Using the Parameterized Geometry application mode you can study how the physics changes when the geometry changes as a function of a parameter. The geometry cannot change dynamically. Rather you have a series of fixed geometries of different shapes. The Parameterized Geometry application mode is available for 2D geometries.

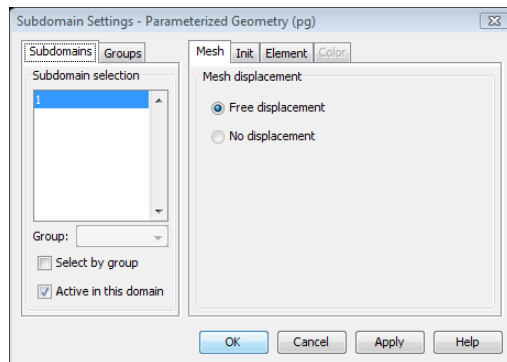
The Parameterized Geometry application mode uses the same technique to calculate the mesh displacement as the Moving Mesh (ALE) application mode. The difference between the two application modes is the available settings on subdomains and boundaries.

See also the “The Moving Mesh Application Mode” on page 463 for more information about deformed meshes. The sections about frames, smoothing methods, and remeshing also applies to the Parameterized Geometry application mode.

## *Subdomain Settings*

---

The **Subdomain Settings** dialog box in this application mode contains properties for the mesh displacement.



*Figure 16-9: The Subdomain Settings dialog box of the Parameterized Geometry application mode.*

On the **Mesh** page you can specify how the displacement of the mesh is computed in each subdomain. The available options are:

- **Free displacement.** The default value. Use this option in domains that change their shapes as a function of the geometrical parameter. The mesh displacement is constrained only by the boundary conditions on the surrounding boundaries. The software obtains the displacement in the subdomain by solving Winslow’s smoothing PDE (see “Smoothing Methods” on page 466).
- **No displacement.** Use this in domains that are not affected by the geometrical parameter.

The **Init** and **Element** pages in the same as in the Moving Mesh (ALE) application mode (see “Subdomain Settings” on page 463).

### Boundary Conditions

To set the boundary conditions, use the **Boundary Settings** dialog box.

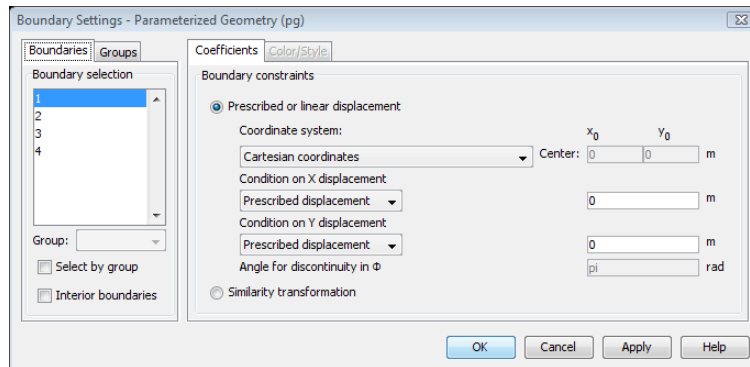


Figure 16-10: The *Boundary Settings* dialog box of the *Parameterized Geometry* application mode.

The boundary conditions in this application mode fall into three categories:

- **Prescribed displacement.** Choose this option to define the displacement as an expression.
- **Linear displacement.** Choose this option when you have prescribed the displacement of the boundary’s end points. The linear displacement then ensures that you get a smooth deformation of the mesh between the end points. The displacement can be a linear function of  $X$ ,  $Y$ , the curve parameter  $s$ , the angular coordinate  $\varphi$ , or the radial coordinate  $r$  (the latter two are only available when you select **Polar**



**coordinates**). For example, if you choose the displacement  $dx = x - X$  to be **Linear in X**, then the software imposes the constraint

$$dx = dx_1 + (dx_2 - dx_1) \frac{X - X_1}{X_2 - X_1}$$

along the boundary, where  $dx = dx_1$  at one end point  $X = X_1$  and  $dx = dx_2$  at the other end point  $X = X_2$ .

- **Free.** Choose this option if you want the boundary to move freely without any constraint on the displacement.
- **Similarity transformation.** Choose this option when you have prescribed the displacements of the boundary's end points. The software translates, rotates, and scales the boundary so that the constraints at the end points are satisfied. If  $dx$  and  $dy$  are the displacements in the  $x$  and  $y$  directions,  $dx_1$  and  $dy_1$  are the values at  $X = X_1$ , and  $dx_2$  and  $dy_2$  the values at  $X = X_2$ , then the similarity transformation is

$$\begin{aligned} dx &= a(X - X_1) - b(Y - Y_1) + dx_1 \\ dy &= b(X - X_1) + a(Y - Y_1) + dy_1 \\ a &= \frac{(dx_2 - dx_1)(X_2 - X_1) + (dy_2 - dy_1)(Y_2 - Y_1)}{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \\ b &= \frac{-(dx_2 - dx_1)(Y_2 - Y_1) + (dy_2 - dy_1)(X_2 - X_1)}{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \end{aligned}$$

For the prescribed and linear-displacement boundary conditions, you can choose condition on the two displacement components independently.

In the **Coordinate system** list you choose the coordinate system for the mesh displacements in the prescribed and linear-displacement boundary conditions:

- **Cartesian coordinates.** Choose this option to specify conditions on the Cartesian mesh displacements  $dx = x - X$  and  $dy = y - Y$ .
- **Polar coordinates.** Choose this option to specify the center point  $(x_0, y_0)$  of a polar coordinate system with radial coordinate  $R$  and angular coordinate  $\Phi$ . You can also specify an angle from the  $x$ -axis where the angular coordinate will be discontinuous. This makes it possible to specify conditions on the radial displacement  $dr = r - R$  and the angular displacement  $d\phi = \phi - \Phi$ , where  $r$  and  $\phi$  are the values of the polar coordinates in the deformed configuration.
- **Tangent and normal coordinate system in reference mesh.** Choose this option to specify conditions on the displacements tangential and normal to the reference

boundary. That is, you can specify conditions on the tangential displacement  $tX*dx+tY*dy$  and the normal displacement  $nX*dx+nY*dy$ .

- **Tangent and normal coordinate system in deformed mesh.** Choose this option to specify conditions on the displacements tangential and normal to the deformed boundary. That is, you can specify conditions on the tangential displacement  $tX*dx+tY*dy$  and the normal displacement  $nX*dx+nY*dy$ .

### *Point Settings*

---

In the **Point Settings** dialog box it is possible to constrain the displacement of points in the geometry. You can choose to specify the displacements in Cartesian or polar coordinates.

# Example—Eigenmode Analysis of an L-Shaped Membrane with Rounded Corner

## *Introduction*

---

Cleve Moler studied the eigenvalues of the L-shaped membrane in his Ph.D. thesis (Ref. 1). Extend the problem by introducing a rounded corner in the concave corner and plot the radius versus the eigenvalue of the rounded corner. COMSOL Multiphysics makes it possible to easily parameterize the problem in terms of the radius of the rounded corner.

## *Model Definition*

---

This model is a study of how the eigenvalues of the eigenvalue problem

$$\begin{cases} -\Delta u = \lambda u & \text{on } \Omega \\ u = 0 & \text{on } \partial\Omega \end{cases}$$

change as a function of the radius of the rounded corner. To use the nonlinear parametric solver you need to add a normalizing condition that determines the size of the solution. In this model, it is the integral condition

$$\int_{\Omega} |u|^2 dx dy = 1$$

## *Modeling in COMSOL Multiphysics*

---

Use an integration coupling variable to compute the norm and an ordinary differential equation (ODE) to enter the equation for the constraint on the norm. The value of the Lagrange multiplier  $\lambda$  corresponds to the eigenvalue of the problem. You can also use this technique to solve nonlinear eigenvalue problems such as  $\Delta u = \lambda \sin(u)$ .

Parameterize the radius of the rounded corner by a parameter  $p$ . Initially draw the geometry with a rounded corner with radius 0.5. Scale the straight boundaries adjacent to the rounded corner in the  $x$  and  $y$  directions, respectively. Scale the rounded corner

by a similarity transformation. Drive the parameterization by introducing point constraints involving the parameter  $p$  in the vertices adjacent to the rounded corner.

Use the COMSOL Multiphysics eigenvalue solver to get the initial values for the parametric study. Use the parametric solver to track the eigenmodes and eigenvectors from a parameter  $p$  from 0 to 0.5. Then, finally, match the original eigenmodes of the L-shaped membrane with the ones derived by the parametric study.

## Results

It is possible to continuously deform the first eigenmodes of the problem with a rounded corner to eigenmodes of the problem with a sharp corner. The 4th mode of the problem with rounded corner can be deformed to the 5th mode of the problem with sharp corner, and vice versa. The 8th and 9th modes of the problem with rounded corner merge into a multiple mode for the problem with the sharp corner.

TABLE 16-1: CORRESPONDENCE BETWEEN EIGENVALUES AND EIGENMODES OF THE PROBLEM WITH ROUNDED AND SHARP CORNERS

EIGENVALUE NUMBER	ROUNDED CORNER, RADIUS = 0.5	SHARP CORNER	MAPPING: ROUNDED TO SHARP
1	8.42	9.80	1→1
2	14.9	15.2	2→2
3	19.6	19.8	3→3
4	28.7	29.7	4→5
5	29.6	32.2	5→4
6	39.4	41.9	6→6
7	43.5	45.4	7→7
8	48.8	49.6	8→(8,9)
9	49.8	49.6	9→(8,9)

## Reference

1. C.B. Moler, “Finite Difference Methods for the Eigenvalues of Laplace’s Operator,” STAN-CS-65-22, Stanford University, 1965.

**Model Library path:** COMSOL\_Multiphysics/Benchmarks/  
lshaped\_membrane\_parametric

#### **MODEL NAVIGATOR**

- 1 Select **2D** from the **Space dimension** list.
- 2 In the application mode list, open **COMSOL Multiphysics>Deformed Mesh** and then **Parameterized Geometry**.
- 3 Click the **Multiphysics** button and then the **Add** button.
- 4 In the application mode list, open **COMSOL Multiphysics>PDE Modes** and then double-click **PDE, Coefficient Form**. Make sure that the **PDE, Coefficient Form** application mode appears together with the **Parameterized Geometry** application mode under **Frame (pg)**.
- 5 Click **OK**.

#### **OPTIONS AND SETTINGS**

- 1 From the **Options** menu, choose **Axes/Grid Settings**.
- 2 Click the **Grid** tab, clear the **Auto** check box, and add **Extra X: -0.5** and **Extra Y: 0.5**.
- 3 Click **OK**.

#### **GEOMETRY MODELING**

- 1 On the **Draw** toolbar, click **Line**.
- 2 Click  $(-1, 0)$ ,  $(-1, -1)$ ,  $(1, -1)$ ,  $(1, 1)$ ,  $(0, 1)$ ,  $(0, 0.5)$ .
- 3 On the **Draw** toolbar, click **2nd Degree Bézier Curve**.
- 4 Click  $(0, 0)$ ,  $(-0.5, 0)$ .
- 5 Close the geometry object by right clicking the mouse.
- 6 Click the **Zoom Extents** button on the Main toolbar.

#### **PHYSICS SETTINGS**

##### *Subdomain Settings*

- 1 Make sure the **PDE, Coefficient Form (c)** application mode is selected on the **Multiphysics** menu.
- 2 On the **Physics** menu, open the **Subdomain Settings** dialog box.
- 3 Select Subdomain 1.
- 4 Enter the source term  $1m*u$  in the **f** coefficient edit field.
- 5 On the **Init** tab, enter the initial value  $u$  in the  **$u(t_0)$**  edit field; then click **OK**.

### *Coupling Variables*

- 1 From the **Options** menu, choose **Integration Coupling Variables>Subdomain Variables** to open the **Subdomain Integration Variables** dialog box.
- 2 On Subdomain 1, define the variable `uInt` as the expression  $u^2$  using integration order 4. Make sure the **Global destination** check box is selected.
- 3 Click **OK**.

### *Global Equations*

- 1 On the **Physics** menu, open the **Global Equations** dialog box.
- 2 On the first row in the table on the **States** tab, type `1m` in the **Name (u)** column and  $1 - uInt$  in the **Equation** column. Set its initial value to `1ambda` in the **Init (u)** column.
- 3 Click **OK**.

### *Boundary Conditions*

- 1 In the **Multiphysics** menu, select the **Parameterized Geometry (pg)** application mode.
- 2 Open the **Boundary Settings** dialog box on the **Physics** menu.
- 3 Keep the default setting on Boundaries 1, 2, 5, and 6.
- 4 On Boundary 3, select **Prescribed or linear displacement, Linear in X for Condition on X displacement**, and **Prescribed displacement, 0, for Condition on Y displacement**.
- 5 On Boundary 4, select **Prescribed or linear displacement, Prescribed displacement, 0, for Condition on X displacement**, and **Linear in Y for Condition on Y displacement**.
- 6 Select **Similarity transformation** for the rounded corner, Boundary 7.
- 7 Click **OK**.

### *Point Settings*

- 1 Open the **Point Settings** dialog box on the **Physics** menu.
- 2 Select Point 3, and check **X-displacement**, and set `p` as its value.
- 3 Select Point 4, and check **Y-displacement**, and set `-p` as its value.
- 4 Click **OK**.

## **POSTPROCESSING AND VISUALIZATION**

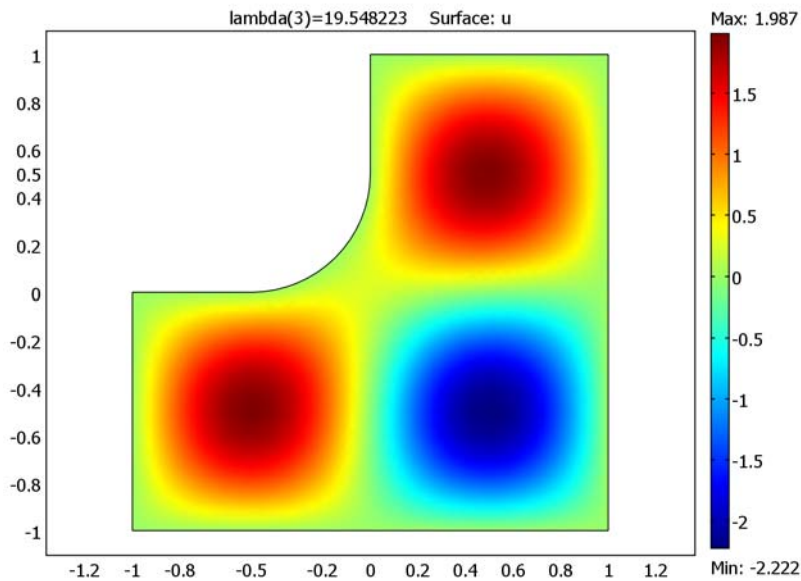
- 1 Open the **Plot Parameters** dialog box.
- 2 Click the **Surface** tab. Select **PDE, Coefficient Form (c)>u** from the **Predefined quantities** list on the **Surface Data** page.
- 3 Click **OK**.

### COMPUTING THE SOLUTION

- 1 Open the **Solver Parameters** dialog box.
- 2 Select the **Eigenvalue** solver.
- 3 Click **OK**.
- 4 Open the **Solver Manager** dialog box.
- 5 On the **Solve For** page, select only the degree of freedom variable  $u$ .
- 6 Click **OK**.
- 7 Click the **Solve** button on the Main toolbar.

### POSTPROCESSING AND VISUALIZATION

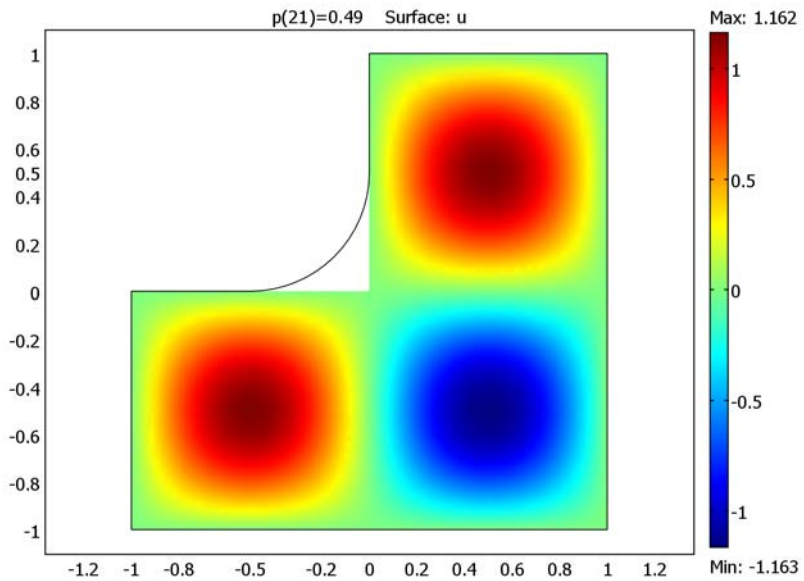
- 1 Open the **Plot Parameters** dialog box.
- 2 On the **General** page, select the third eigenvalue (approximately 19.5) from the list.
- 3 Click **OK**.



### COMPUTING THE SOLUTION

- 1 Open the **Solver Manager** dialog box.
- 2 On the **Initial Value** page, make sure that the **Initial value expression evaluated using current solution** button is selected.

- 3 Select the third eigenvalue from the **Eigenvalue** list.
- 4 On the **Solve For** page, press Ctrl+A to select all variables:  $u$ ,  $1m$ ,  $x$ , and  $y$ .
- 5 Click **OK**.
- 6 Open the **Solver Parameters** dialog box.
- 7 Select the **Parametric** solver.
- 8 Set the **Parameter name** to  $p$ , and the **Parameter values** to `linspace(0,0.49,21)`.
- 9 Click **OK**.
- 10 Click the **Solve** button on the Main toolbar.



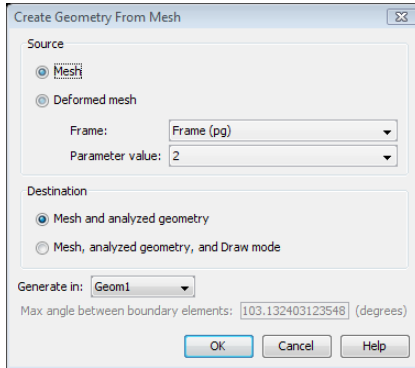
### POSTPROCESSING AND VISUALIZATION

Click the **Animate** button on the Plot toolbar to see an animation of the parameterization.



# Retrieving the Deformed Mesh

You can retrieve a deformed mesh by selecting **Create Geometry From Mesh** from the **Mesh** menu.



*Figure 16-11: The Create Geometry From Mesh dialog box.*

To retrieve a deformed mesh, select **Deformed Mesh** in the **Source** frame. The **Frame** and **Parameter value** or **Solution at time** drop-down lists let you select which source frame and solution to retrieve the deformed mesh from.

In the **Destination** frame you can select if you want to create a mesh and an analyzed geometry or a mesh, an analyzed geometry and a Draw mode object from the deformed mesh. Use the **Generate in** drop-down list to select which geometry to put the deformed mesh in.

In the **Max angle between elements in planar face** edit field you specify the maximum allowed angle between two boundary elements to be part of the same planar face. This parameter is only available if the mesh was initially imported and has no parametrization.

# Limitations of the Deformed Mesh Feature

The following limitations apply to the deformed mesh feature:

- The connectivity of the mesh remains unchanged during the mesh deformation, which means that topological changes in the geometry are not allowed.
- When the mesh deformation becomes large, the quality of the mesh can deteriorate. If this happens, the solver runs into convergence problems. Sometimes you see the warning **Inverted mesh element** in the Progress window for the solver, which means that a mesh element has (partially) warped inside-out. Use a smaller mesh deformation if you get these problems.
- If you work with finite elements of types other than Lagrange, solving for the mesh deformation coupled to the non-Lagrange elements can result in convergence problems. However, solving first for the mesh deformation and then for the physics in a one-way coupled fashion works. The reason for this is that if  $u$  is discretized with non-Lagrange elements, first-order spatial derivatives like  $u_x$  does not give a contribution to the Jacobian matrix from the dependence on the mesh motion. Moreover, second-order spatial derivatives never make such Jacobian contributions.
- Solving time-dependent wave equations on a moving mesh does not work, because the formula for the second-order spatial time derivative  $u_{tt}$  in terms of reference time derivatives is not implemented.
- When using second-order (or higher-order) elements in the deformed mesh application modes, the mesh moving techniques often produce elements with distorted shapes. The measure of mesh quality does not capture these distorted shapes because it is computed from the positions of the corners of the mesh element (ignoring mid-side nodes, for instance). For these reasons, it is often best to use linear elements for the mesh positions in the deformed mesh application modes. You can select element type on the **Element** tab in the **Subdomain Settings** dialog box for the Moving Mesh (ALE) and Parameterized Geometry application modes. You must also open the **Model Settings** dialog box from the **Physics** menu and select the ALE frame (typically **Frame (ale)**) in the **Frames** list and select the appropriate element order (typically **Linear**) in the **Geometry shape order** list.

## Stabilization Techniques

Numerical solutions of transport equations can sometimes exhibit oscillations even when the exact solutions are smooth. These spurious oscillations are caused by numerical instabilities.

The first section of this chapter describes when spurious oscillations can arise. To restore numerical stability COMSOL Multiphysics provides a set of stabilization techniques, which are described in the second section.

This chapter does not contain description of the user interface which is instead described in conjunction with each application mode that supports any stabilization technique.

# Numerical Stabilization

This chapter discusses the numerical stability of the generic scalar convection-diffusion transport equation

$$\frac{\partial u}{\partial t} + \beta \cdot \nabla u = \nabla \cdot (c \nabla u) + F \quad (17-1)$$

where  $\beta$  is the convective velocity vector,  $c$  is the diffusion coefficient,  $u$  is a transported scalar, and  $F$  is a source term. The underlying finite element discretization method in COMSOL Multiphysics is the Galerkin method. When discretizing Equation 17-1 using the Galerkin Method, it is well known that the resulting numerical problem becomes unstable for an element Péclet number (Pe) larger than one (Ref. 1):

$$\text{Pe} = \frac{\|\beta\| h}{2c} > 1 \quad (17-2)$$

where  $h$  is the mesh element size. The Péclet number is a measure of the relative importance of the convective effects compared to the diffusive effects; a large Péclet number indicates that the convective effects dominate over the diffusive effects.

Oscillations can occur where any of the following conditions exist and the Péclet number exceeds one:

- A Dirichlet boundary condition can lead to a solution containing a steep gradient near the boundary, forming a boundary layer. If the mesh cannot resolve the boundary layer, this creates a local disturbance.
- A space-dependent initial condition that the mesh does not resolve can cause a local initial disturbance that propagates through the computational domain.
- A small initial diffusion term close to a non-constant source term or a non-constant Dirichlet boundary condition can result in a local disturbance.

As long as diffusion is present, there is—at least in theory—a mesh resolution beyond which the discretization is stable. This means that the spurious oscillations can be removed by refining the mesh. In practice, this method is seldom feasible because it can require a very dense mesh. Instead, it is common practice to use stabilization methods, that is, methods that adds artificial diffusion. COMSOL Multiphysics provides several of these methods and they are described in the next section.

### Stabilization—An Example

To illustrate the concepts of this chapter, consider the problem

$$\cos\left(\frac{\pi}{3}\right)\frac{\partial u}{\partial x} + \sin\left(\frac{\pi}{3}\right)\frac{\partial u}{\partial y} = 10^{-4}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) + 1 \quad (17-3)$$

solved on the unit square. Equation 17-3 is discretized using 10 times 10 biquadratic Lagrangian elements. The boundary conditions are:

- $u = 1$  for  $x = 0$
- $u = 1$  for  $y = 0$
- $u = 0$  for  $x = 1$
- $u = 0$  for  $y = 1$

Figure 17-1 shows the mesh and boundary conditions. In general, using uniform meshes for transport problems is not recommended. Nevertheless, this example uses a uniform mesh to demonstrate the different stabilization techniques.

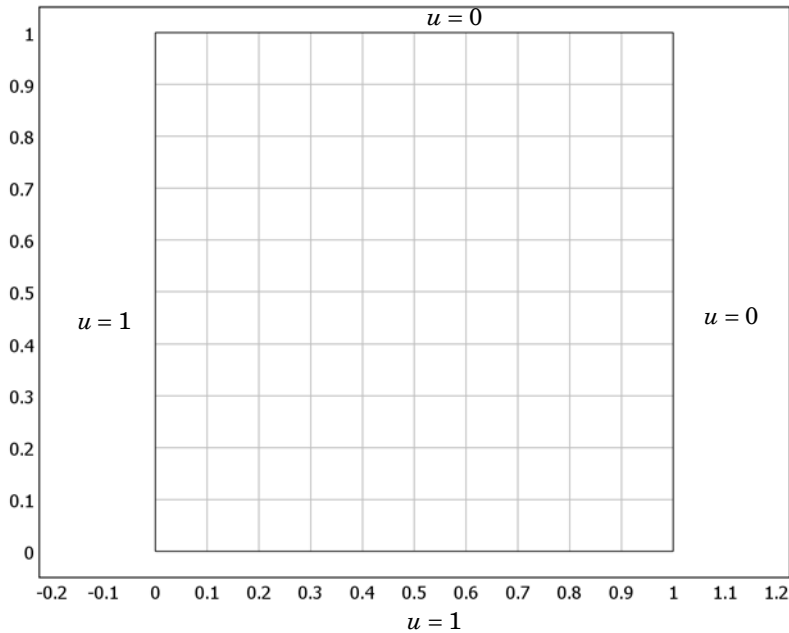


Figure 17-1: The computation domain, mesh and boundary condition for Equation 17-3.

The expected solution rises slowly and smoothly from the left and lower boundaries, and has sharp boundary layers along the upper and right boundaries. Figure 17-2 shows a reference solution obtained using 100-by-100 biquadratic Lagrangian elements, Galerkin least-squares, and crosswind diffusion (see the next section). The arrows indicate the direction of  $\beta$ .

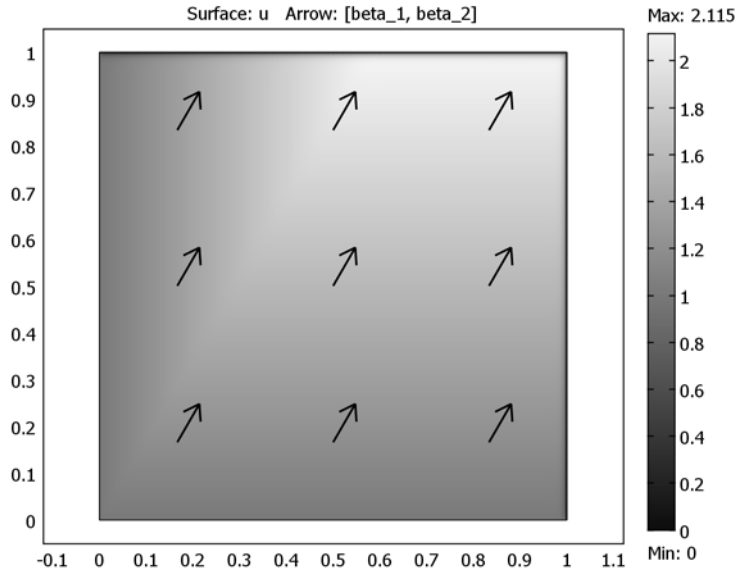


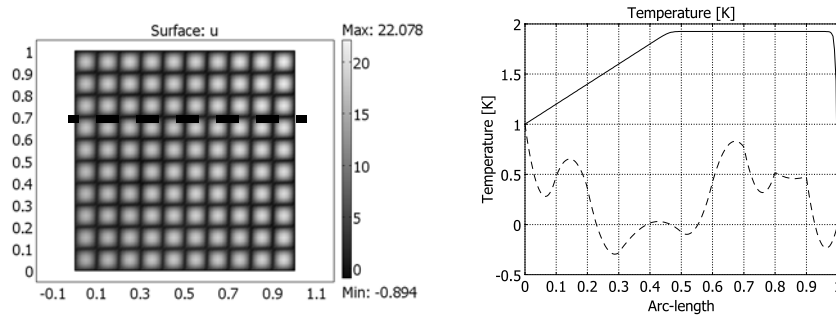
Figure 17-2: Reference solution of Equation 17-3. Solved using 100 times 100 biquadratic elements, GLS, and crosswind diffusion.

The cell Péclet number for this example is

$$Pe = \frac{1 \cdot 0.1}{2 \cdot 10^{-4}} = 500 \gg 1$$

Figure 17-3 displays the solution obtained using the mesh shown in Figure 17-1 and (unstabilized) Galerkin discretization. As can be expected with such a high Péclet number, the unstabilized solution shows little, if any, resemblance to the reference solution in Figure 17-2. The right plot in Figure 17-3 shows a cross-sectional plot

along the dashed line,  $y = 0.8$  and the corresponding reference solution. Notice that the unstabilized solution is completely destroyed by oscillations.



*Figure 17-3: Equation 17-3 solved using unstabilized Galerkin formulation. The right plot compares the unstabilized solution (dashed line) along the dashed line in the left plot ( $y = 0.8$ ) with the reference solution (solid line).*

The next section explores how different stabilization techniques affect the solution of this example.

---

**Note:** The example uses the General Heat Transfer application mode in the Heat Transfer Module.

---

# Artificial Diffusion Types

Several techniques for handling numerical instabilities without the need for mesh refinement are available. They all have in common that they add terms to the transport equation. These terms introduce *artificial diffusion* (artificial viscosity or numerical diffusion/viscosity are also common names) that stabilize the solution. The stabilization methods in COMSOL Multiphysics are described below.

## *Isotropic Diffusion*

---

Adding isotropic diffusion is equivalent to adding a term,

$$c_{\text{art}} = \delta_{\text{id}} h \|\beta\|,$$

to the physical diffusion coefficient,  $c$ . Here,  $\delta_{\text{id}}$  is a tuning parameter. This means that you do not solve the original problem, Equation 17-1, but rather, the modified  $O(h)$  perturbed problem

$$\frac{\partial u}{\partial t} + \beta \cdot \nabla u = \nabla \cdot ((c + c_{\text{art}}) \nabla u) + F$$

With the default value of  $\delta_{\text{id}} = 0.5$ , the new cell Péclet number can be expressed as

$$\text{Pe} = \frac{h \|\beta\|}{2(c + c_{\text{art}})} = \frac{h \|\beta\|}{2c + h \|\beta\|}$$

Clearly, as  $\|\beta\|$  approaches infinity,  $\text{Pe}$  approaches, but never exceeds, one.

While a solution obtained with isotropic diffusion might not be satisfactory in all cases, the added diffusion definitely dampens the effects of oscillations and impedes their propagation to other parts of the system. It is not always necessary to set  $\delta_{\text{id}}$  as high as 0.5 to get a smooth solution, and it should be chosen smaller if possible.



Figure 17-4 shows the effect of isotropic diffusion on Equation 17-3 with  $\delta_{id} = 0.5$ . Although the solution is smooth, the comparison with the reference solution in the right plot reveals that the isotropic diffusion introduces far too much diffusion.

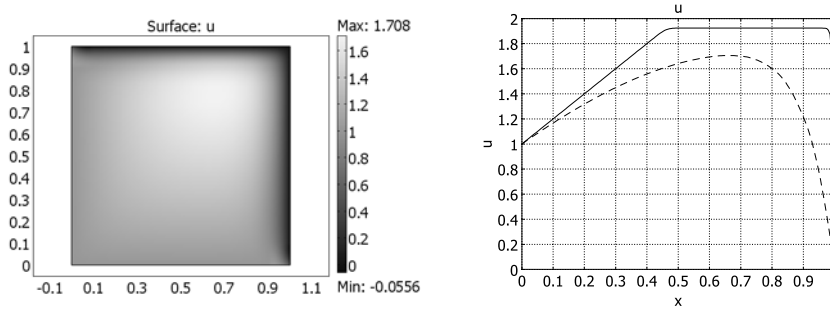


Figure 17-4: Equation 17-3 solved using isotropic diffusion. The right plot compares the stabilized solution (dashed line) along  $y = 0.8$  with the reference solution (solid line).

### Streamline Diffusion

Most application modes that support streamline diffusion support one or several of the following three types:

- Anisotropic diffusion
- Streamline upwind Petrov-Galerkin (SUPG)
- Galerkin least-squares (GLS)

The rest of this section describes each of these methods. For theoretical details, see Ref. 1 and Ref. 2.

#### ANISOTROPIC DIFFUSION

Anisotropic diffusion is a direct refinement of the isotropic diffusion method described above. In many cases, there is no need for any artificial diffusion in the direction orthogonal to  $\beta$ . It is therefore possible to modify  $c_{art}$  to a form that only adds artificial diffusion in the direction of the flow (the streamline direction).

The normal interpretation of the diffusivity in Equation 17-1 is as a scalar number, but it can in the general case be a tensor. This implies that you can also define the coefficient of artificial diffusion as a tensor. The natural choice is to define  $c_{art}$  as

$$c_{art,ij} = \frac{\delta h \beta_i \beta_j}{\|\beta\|}$$

where  $\delta$  is a tunable parameter that controls the amount of artificial diffusion. The value  $\delta = 0.5$  guarantees that the cell Péclet number measured in the streamline direction does not exceed one (compare to isotropic diffusion). Choose  $\delta$  smaller than 0.5 if possible. A good rule of thumb is to select  $\delta = 0.5/p$  where  $p$  is the order of the basis functions.

Analogous to the case of isotropic artificial diffusion, a problem stabilized with anisotropic diffusion is not identical to the original problem. In fact, the exact solution to the original equations does not solve the stabilized problem. However, the errors introduced by the anisotropic artificial diffusion are far less serious than the ones created by isotropic artificial diffusion.

Figure 17-5 displays the effect of anisotropic diffusion on the solution of Equation 17-3 with  $\delta = 0.25$ . The stabilized solution curve closely follows the reference solution up to the sharp boundary layers where oscillations occur. Choosing higher values of  $\delta$  gives a smoother solution. For this simple example, it is possible to get very close to the solution obtained using SUPG by selecting  $\delta = 0.5$ . In general, anisotropic streamline diffusion can at best give solutions close to those obtained by SUPG, but never better.

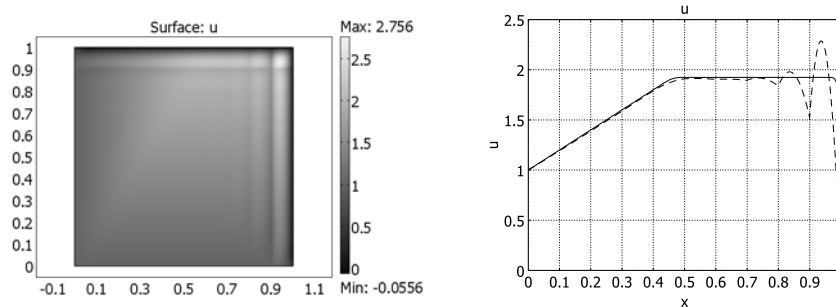


Figure 17-5: Equation 17-3 solved using anisotropic diffusion. The right plot compares the stabilized solution (dashed line) along  $y = 0.8$  with the reference solution (solid line).

### STREAMLINE UPWIND PETROV-GALERKIN (SUPG)

The Streamline upwind Petrov-Galerkin method is a consistent method, which means that it does not perturb the original transport equation (Equation 17-1). A model that converges with this method can be considered to be a solution to the discrete counterpart of Equation 17-1. It is closely related to upwinding schemes in finite difference and finite volume methods.

SUPG can be shown to add a smaller amount of stability than anisotropic diffusion (see Ref. 3). But while the accuracy of anisotropic diffusion is at best  $O(h)$ , the accuracy of SUPG can be shown to be at least  $O(h^{p+1/2})$  where  $p \geq 1$  is the order of the basis functions.

Compared to Galerkin least-squares (GLS), SUPG is, on the one hand, less sophisticated and thereby less stabilizing; on the other hand, SUPG involves fewer discrete terms and is therefore computationally less expensive.

Figure 17-6 displays the effect of SUPG on the solution of Equation 17-3. The solution closely follows the reference solution away from the boundary layers, but at the boundary layers, oscillations occur. This is a typical behavior for streamline diffusion: the solution becomes smooth and exact in smooth regions but can contain oscillations at sharp gradients.

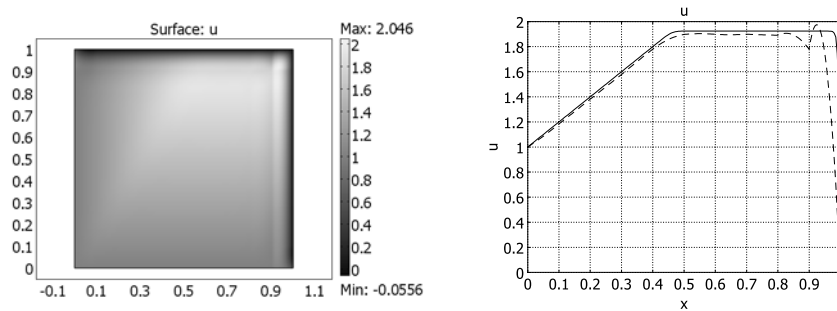


Figure 17-6: Equation 17-3 solved using streamline upwind Petrov-Galerkin. The right plot compares the stabilized solution (dashed line) along  $y = 0.8$  with the reference solution (solid line).

### GALERKIN LEAST-SQUARES (GLS)

Galerkin least-squares (GLS) is a more advanced version of SUPG, with which it shares many features. GLS, for example, is also a consistent method and has the same order of accuracy as SUPG.

To understand the differences between GLS and SUPG, consider the following extended form of Equation 17-1:

$$\frac{\partial u}{\partial t} + \beta \cdot \nabla u = \nabla \cdot (c \nabla u) + s u + F \quad (17-4)$$

where  $s$  is a production coefficient if  $s > 0$  and an absorption coefficient if  $s < 0$ .

If the order,  $p$ , of the basis functions is greater than one, GLS introduces superviscosity, that is, higher-order terms that provide extra stability. GLS and SUPG coincide if  $s = 0$  and  $p = 1$ . If  $s \neq 0$ , the numerical solution of Equation 17-4 is characterized by the Péclet number (see Equation 17-2) and the element Damköhler number:

$$\text{Da} = \frac{|s|h}{\|\beta\|}$$

A new dimensionless number can be formed by combining the Damköhler number and the Péclet number:

$$2\text{DaPe} = \frac{|s|h^2}{c} \quad (17-5)$$

The (unstabilized) Galerkin discretization becomes unstable if  $2\text{DaPe} > 1$  (Ref. 4), that is, if the production/absorption effects dominate over the viscous effects. GLS differs from SUPG in that GLS relaxes this requirement while SUPG does not.

While GLS is in some cases superior to SUPG, it is also computationally more expensive.

The effect of GLS on the example model (Equation 17-3) is almost exactly that given by SUPG. The difference between the methods is visible only in more advanced examples or if  $|s| \gg 1$ .

### *Crosswind Diffusion*

---

Streamline diffusion introduces artificial diffusion in the streamline direction. This is often enough to obtain a smooth numerical solution provided that the exact solution of Equation 17-1 (or Equation 17-4) does not contain any discontinuities. At sharp gradients, however, undershoots and overshoots can occur in the numerical solutions. Crosswind diffusion addresses these spurious oscillations by adding diffusion orthogonal to the streamline direction, that is, in the crosswind direction.

Most crosswind diffusion methods are consistent; that is, they do not alter the equation. The most efficient methods are nonlinear. This means that the discrete equation system becomes nonlinear even if the original equation (Equation 17-1 or Equation 17-4) is linear, which can increase the computational cost.

Use crosswind diffusion if it is important to avoid undershoots or overshoots. Typical examples are concentrations that must not become negative and mass fractions that must not be greater than one.

Ref. 5 presents a comprehensive review of most of the existing crosswind diffusion methods.

Figure 17-7 shows the example problem (Equation 17-3) solved using SUPG and crosswind diffusion. Oscillations at the boundary layers are almost completely removed (compare with Figure 17-6), but it has been achieved by the introduction of some extra diffusion. In general, crosswind diffusion tries to smear out the boundary layer so that it becomes just wide enough to be resolved on the mesh (Figure 17-1). To obtain a sharper solution and remove the last oscillations, the mesh needs to be refined locally at the boundary layers.

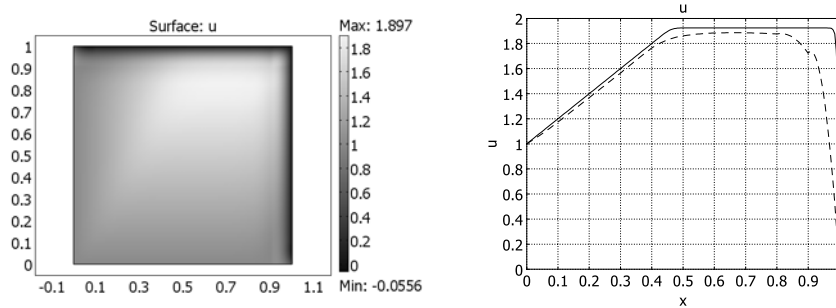


Figure 17-7: Equation 17-3 solved using streamline upwind Petrov-Galerkin and crosswind diffusion. The right plot compares the stabilized solution (dashed line) along  $y = 0.8$  with the reference solution (solid line).

## References

1. O.C. Zienkiewicz, R.L. Taylor, and P. Nithiarasu, *The Finite Element Method for Fluid Dynamics*, 6th edition, Elsevier, 2005.
2. R. Codina, “Comparison of some finite element methods for solving the diffusion-convection-reaction equation,” *Comput. Methods Appl. Mech. Engrg.*, vol. 156, pp. 185–210, 1998.
3. C. Johnson, *Numerical solution of partial differential equations by the finite element method*, Studentlitteratur, 1987.

4. G. Hauke, “A simple subgrid scale stabilized method for the advection-diffusion-reaction equation,” *Comput. Methods Appl. Mech. Engrg.*, vol. 191, pp. 2925–2947, 2002.
5. V. John and P. Knobloch, “On spurious oscillations at layers diminishing (SOLD) methods for convection-diffusion equations: Part I - a review,” *Comput. Methods Appl. Mech. Engrg.*, vol. 196, pp. 2197–2215, 2007.

# I N D E X

- A** absorption coefficient 267
- absorption term 254
  - variable for 269
- AC Power Electromagnetics application mode 107
- AC/DC Module 78
- accuracy of results 9
- acoustics
  - application mode for 23
  - boundary conditions for 21
  - mathematical models for 21
  - standard problems in 20
- active application mode 388
- Add Frame dialog box 459
- adjoint method, vs. forward method 306
- adjoint sensitivity 296
- adsorption, modeling using the weak form 368
- ALE method 454
- algebraic equations 346
- a1u boundary coupled equation variable 271
- a1u equation variable 269
- Ampère's law 79, 107
- analysis
  - eigenfrequency 208
  - static 208
  - time-dependent 208
  - transient 208
- analysis types, in structural mechanics 207
- anisotropic diffusion 501
- anisotropic material 253
- antisymmetry conditions 7
- application mode
  - AC Power Electromagnetics 107
  - Acoustics 23
  - active 388
  - Axial Symmetry, Stress-Strain 230
  - Conduction 171
  - Conductive Media DC 85
  - Convection and Conduction 179
  - Convection and Diffusion 51
  - Diffusion 46
  - Electrostatics 91
  - for physics 15
  - for weak form 351
  - Magnetostatics 97
  - Plane Strain 225
  - Plane Stress 220
  - Poisson's equation 284
  - ruling 388
  - Solid, Stress-Strain 216
- application mode names
  - for PDE modes 246
- application mode properties
  - changing 392
- application mode variables 17
  - for fluid mechanics 148
  - for PDE modes 277
- application modes 2
  - in COMSOL Multiphysics 2
  - removing from model 387
  - selecting 5
- arbitrary Lagrangian-Eulerian method 454
- area effects, across frames in ALE models 461
- artificial diffusion 500
  - for convective heat transfer 181
  - for the Navier-Stokes equations 52
  - using pressure stabilization 496

- artificial diffusion method
  - crosswind diffusion 504
  - isotropic diffusion 500
  - streamline diffusion 501
- artificial viscosity 500
- assemblies, meshing 427
- assembly 414
- au boundary coupled equation variable 271
- au equation variable 269
- average temperature 363
- axial symmetry 7
  - in acoustics 33
  - in Conductive Media DC models 89
  - in diffusion 48
  - in diffusion and convection 55
  - in Electrostatics models 94
  - in heat transfer 175
- Axial Symmetry, Stress-Strain application mode 230
- axisymmetric solution, mapping to 3D 197
- B** Babuska-Brezzi condition 145
- backstep model 154
- benchmark models 9
- benchmark problem 154
- beu boundary coupled equation variable 271
- beu equation variable 269
- body forces 207
- boundary absorption coefficient 267
- boundary absorption term 256
- boundary conditions
  - continuity 175
  - Dirichlet 248
  - for parameterized geometries 484
  - for PDEs 248
  - generalized Neumann 248
  - heat flux discontinuity 176
  - Neumann 248
  - specification of 14
  - symmetric 174
  - thin thermally resistive layer 176
  - with time derivatives 279
- boundary coupled equation variables 270, 271
- boundary layer 425
- boundary source term 256, 267
- bulk modulus 21
- C** cell Reynolds number 141
- CFD 134
- CFL number 150
- Chemical Engineering Module
  - Convection and Diffusion application mode in 51
  - diffusion application modes in 46
  - for fluid dynamics 134
- circumferential wave number 33
- classical PDEs 268
- Classical PDEs application mode
  - Poisson's equation 284
- coefficient form 5, 264
- complex-valued variables, in sensitivity analysis 295
- Component Library 409
- components 409
- composite geometry 414
- computational fluid dynamics 134
- concentration
  - boundary condition for 48
  - of mass 46
- Conduction application mode 171
- conduction, in heat transfer 168
- Conductive Media DC
  - application mode 85
- conductivity 80



- in Joule heating 86
  - temperature-dependent 86, 401
  - thermal 173
- conductor, perfect 83
- connecting geometries, model example 442
- conservative convective flux
  - for two-variable PDE 253
  - normal component of 255
  - variables for 269
- conservative flux 267
  - source term 254
- conservative flux convection coefficient 267
- conservative flux source
  - normal component of 256
- conservative flux source term 267
- conservative formulation
  - for convection and diffusion 51
- constitutive relations
  - generalized 81
  - in electromagnetics 80
- constraints
  - general notation for 214
  - ideal 273
  - ideal vs. non-ideal 359
  - non-ideal 273, 274
  - specification of 213
- contact pair 417
- contact resistance 88, 425
- contiguous memory block 8
- continuity 176
- continuity boundary condition 175
- Convection and Conduction
  - application mode 179
- Convection and Diffusion
  - application mode 51
- convection coefficient 267
  - convection term
    - for two-variable PDE 254
    - variable for 269
  - convection, in heat transfer 168
  - convection-diffusion equation 268
  - convective flux, boundary condition for 185
  - convergence
    - for nonlinear equations 10
    - testing 9
  - coordinate system 455
  - copper, resistive heating of 395
  - cost function 294, 314
  - coupled-field analysis 376
  - coupling operator 420
  - coupling variable
    - integration 487
  - coupling, in resistive heating model 395, 400
  - crosswind diffusion 146, 504
  - crosswind diffusion stabilization 183
  - cu boundary coupled equation variable 271
  - cu equation variable 269
  - current density 79
  - current flow 87
  - current, steady 85
  - cut-on frequency 37
  - cylindrical coordinates 230
  - cylindrical wave 27
- D** DAE. See differential-algebraic equation.
  - da1u boundary coupled equation variable 271
  - damping 215
  - damping coefficient 267
  - damping term 249, 252
  - dau boundary coupled equation variable 271

- dau equation variable 269
- dbeu boundary coupled equation variable 271
- dcu boundary coupled equation variable 271
- deactivating physics or equations 391
- deformed mesh 454
  - limitations 494
  - minimum quality of 468
- deformed mesh application mode, adding 459
- density 169
  - in fluids 135
  - in structural mechanics 212
- dependent variables, editing names of 392
- df boundary coupled equation variable 272
- dga boundary coupled equation variable 271
- dielectric conductor 83
- differential-algebraic equation 266
- Diffusion application mode 46
- diffusion coefficient 46, 267
- diffusion, thin-layer model 442
- diffusive flux 185
  - for two-variable PDEs 252
  - normal component 255
  - variables for 269
- dipole source 21
- Dirac function 286
- Dirichlet boundary condition 248
  - for two-variable PDE 255
- discontinuity in fields 425
- discontinuous flux, across interior boundaries 250
- discontinuous Galerkin method 280
- distorted shapes of mesh elements 494
- distributed resistance 87
- divergence operator 254
- dna1u boundary coupled equation variable 272
- dncu boundary coupled equation variable 272
- dnga boundary coupled equation variable 272
- documentation, for the physics modes 16
- dynamic viscosity 40, 135, 162

**E**

- eigenfrequency analysis 208
- eigenvalue 208
- eigenvalue analysis, in acoustics 23
- eigenvalue problem 266, 487
  - nonlinear 487
- electric charge density 79
- electric conductivity 80, 400, 401
- electric current 400
- electric dipole moment 80
- electric displacement 79
- electric field 79
- electric flux density 79
- electric insulation 87, 99, 109
- electric polarization 80
- electric potential, boundary condition for 87, 93
- electric resistance 400
- electric scalar potential 81
- electric sensor, model of 121
- electric susceptibility 80
- electrochemical polishing 476
- electromagnetic material properties 81
- electromagnetic forces 83, 84
- electromagnetics, fundamental quantities in 79
- Electrostatics application mode 91
- electro-thermal interaction 380
- element Péclet number 496

- element types, selecting 9
  - equation of continuity, in electromagnetics 79
  - equation system
    - applications for 393
    - viewing 393
  - equation system view 2
  - equation variables 268
  - equation-based modeling 282
  - equations
    - deactivating in subdomains 391
  - equilibrium equations 207
  - essential boundary condition 248
  - Eulerian method 454
  - Euler-Lagrange equation 354
  - extended multiphysics 392
  - external current 81
- F**
- f boundary coupled equation variable 272
  - f equation variable 269
  - Faraday's law 79
  - fluid mechanics 134
  - fluid-chemical reactions interaction 381
  - fluid-structure interaction 382
  - fluid-thermal interaction 382
  - flux 267
    - conservative 267
    - vector 258
  - flux discontinuity 48, 55
  - flux vector 250
  - forward method, vs.adjoint method 306
  - forward sensitivity 296
  - Fourier's law of heat conduction 173
  - frames
    - attaching application modes to 461
    - for deformed meshes 458
    - selecting for coupling variables 462
    - settings for 462
    - types of 455
    - when using remeshing 467
  - free boundary 465
  - free displacement 463, 484, 485
  - frequency, in AC Power Electromagnetics models 108
- G**
- g equation variable on boundary 269
  - ga boundary coupled equation variable 271
  - ga equation variable 269
  - Galerkin least-squares stabilization 503
  - Galerkin least-squares streamline diffusion 145
  - Gauss' formula 368
  - Gauss' law 79, 121
  - general form 5, 259
    - for PDEs 258
    - of time-dependent problem 264
  - generalized Neumann boundary conditions 248
    - for coefficient form PDE 255
  - geometries
    - adding to model 392
    - memory-efficient modeling of 8
  - geometry
    - assembly 414
    - composite 414
    - parameterized 487
  - global equations 344
  - goal function 294, 314
  - gradient, of temperature 173
  - gradients 292
  - gravity load 240
  - Green's formula 368
  - ground potential 88
  - ground, boundary condition for 93
- H**
- heat capacity 169, 171, 173, 402
  - heat equation 169, 171, 268

- heat flux 173
  - boundary condition for 174
- heat flux discontinuity 176
- heat sink 169
- heat source 169, 173, 400, 402
- heat transfer
  - application modes for 167
  - by conduction 168
  - by convection 168
  - by radiation 168
  - model examples 188
- heat transfer coefficient 174
  - for transversal convective heat transfer 172
- Heat Transfer Module 168
- Helmholtz equation 267, 268
- Hooke's law 212
- hysteresis losses 114
- I**
  - ideal boundary constraint 364
  - ideal constraints 273, 359, 370
    - in identity conditions 440
  - identity boundary conditions 438
    - example using 442
    - modeling procedure using 440
  - identity pairs 415, 418
    - tips for modeling using 420
  - imprints 415, 428
    - when meshing 427
  - incompressible flow 134
    - modeling 154
  - Incompressible Navier-Stokes
    - application mode 159
  - induction heating 381
  - initial values 403
    - specifying 388
  - integral constraints 318
  - integration coupling variable 487
    - for computing average temperature 363
- interface conditions
  - for Conductive Media DC models 86
- interior boundaries 427
- inward current flow, boundary condition for 87
- isotropic artificial diffusion 500
- isotropic diffusion 146, 500
- isotropic diffusion stabilization 184
- isotropic material 206
- J**
  - Joule heating 86, 380, 395
- K**
  - Karush-Kuhn-Tucker conditions 305
  - Khan and Richardson force 152
- L**
  - L2 error norm 290
  - Lagrange multipliers 366
    - in boundary conditions 249
    - variable names for 359
  - Lagrange p2-p1 elements 156
  - Lagrangian method 454
  - Laplace operator 248
  - Laplace smoothing 466
  - Laplace's equation 268
  - line charge, in Electrostatics models 94
  - line source, in Conductive Media DC models 89
  - linear displacement 484
  - linear problem 249
  - Lorentz force 84
  - Lotka-Volterra equation 345
  - L-shaped membrane 487
- M**
  - magnet, permanent 126
  - magnetic dipole moment 80
  - magnetic field 79
    - boundary condition for 99, 109
  - magnetic flux density 79
  - magnetic hysteresis losses 114
  - magnetic insulation 99, 110

- magnetic potential, boundary condition
    - for 99, 109
  - magnetic susceptibility 80
  - magnetic vector potential 81
  - magnetization 80
  - Magnetostatics application mode 97
  - mass coefficient 249, 264, 267
  - mass damping 215
  - mass matrix 249, 264
  - mass term 252
  - mass transfer coefficient 48
  - material properties, specification of 14
  - material time derivative 457
  - Maxwell's equations 79
  - Maxwell-Ampère's law 79
  - memory use, for COMSOL Multiphysics
    - models 8
  - mesh frames 468
  - mesh quality, stopping when poor 468
  - mesh time derivative 456
  - meshing assemblies 427
  - meta variables, for test functions 274
  - microwave heating 381
  - minimal surface 354
  - minimal-surface 354
  - minimum quality of deformed mesh 468
  - mixed boundary condition 248
  - modeling approaches for multiphysics
    - 376
  - modulus of elasticity 206
  - moving boundaries 454, 476
  - moving mesh 454
  - Moving Mesh application mode 463
  - multigeometry linking, modeling procedure 441
  - multiphysics modeling, combining application modes for 6
  - multiphysics models
    - resistive heating 395
    - specifying settings for 393
- N**
- NAFEMS, benchmark models from 188, 235
  - na1u boundary coupled equation variable 272
  - natural boundary condition 248
  - Navier's equation 207, 208
  - Navier-Stokes equations 134
  - ncu boundary coupled equation variable 272
  - Neumann boundary conditions 248, 260
    - for coefficient form PDE 255
    - generalized 255
  - Newton's second law 208
  - nga boundary coupled equation variable 272
  - no displacement 464, 484
  - nonconservative formulation, for convection and diffusion 51
  - non-ideal constraints 273, 274, 359
  - nonlinear eigenvalue problem 487
  - nonlinear equations, solving 10
  - nonlinear problem 249
  - non-Newtonian fluid 135
  - normal acceleration 27
  - normal components
    - of conservative convection term 255
    - of conservative source term 256
    - of diffusive flux 255
  - normal heat flux continuity 176
  - normal stress 205
  - numerical diffusion 500
  - numerical viscosity 500
- O**
- objective function 314
  - objective functions 294, 317
    - point contributions to 322
  - objective functions, for frequency-do-

- main modeling 295
- ODEs
  - implementing integral constraints 487
  - on boundaries 347
  - solving 344
- Ohm's law 85
- optimization variable 314
- optimization variables
  - specifying 320
- optimization, solver settings for 326
- ordinary differential equations 344
- oscillations in the solution 496
- out-of-core solver, PARDISO 8

**P**

- p2-p1 elements 156
- pair
  - contact 417
  - creating 415
  - identity 415
- parameterized geometry 487
- Parameterized Geometry application
  - mode 483
- PARDISO out-of-core solver 8
- particle charge 84
- particle Reynolds number 152
- particle tracing 83
- PDE
  - classical 268
  - coefficient names for 267
  - on boundaries, edges, and points 352
- PDE formulation, coefficient form vs.
  - general form 263
- PDE modes 2
- PDE system 259
- PDE-constrained optimization 315
- Péclet number 496
  - local 500
- perfect conductor 83
- permanent magnet 126
- permeability 126
  - anisotropic 80
  - of vacuum 80, 98
  - relative 80
- permittivity
  - anisotropic 80
  - of vacuum 80
  - relative 80
- permittivity of vacuum, in electrostatics
  - 92
- physics induced displacement 463
- physics mode
  - AC Power Electromagnetics 107
  - Acoustics 23
  - Conduction 171
  - Conductive Media DC 85
  - Convection and Conduction 179
  - Convection and Diffusion 51
  - Diffusion 46
  - Electrostatics 91
  - Magnetostatics 97
- physics modes 2, 15
- physics, deactivating in subdomains 391
- Plane Strain application mode 225
- Plane Stress application mode 220
- plane stress, model using 235, 240
- plane wave 27
- point contributions to objective function
  - 322
- point settings, specification of 14
- point source 283, 285
  - in AC Power Electromagnetics models
    - 110
  - in Conductive Media DC models 89
  - in Electrostatics models 94
  - in Magnetostatics models 100
- pointwise constraints 319
- Poisson's equation 268, 282

- application mode 284
  - on unit disk 282
- Poisson's ratio 206, 212
- polarization 80
- poroelasticity 386
- predefined multiphysics couplings 376
- prescribed displacement 464, 484
- pressure 135
  - reference value for 29
- pressure source 26
- pressure stabilization
  - Petrov-Galerkin 496
- Q**
  - qu equation variable on boundary 269
  - quantity of interest 294, 314
- R**
  - radiation
    - in acoustics 26
    - in heat transfer 168
  - Rayleigh damping 209, 215
  - reaction force 211
  - reaction rate 47
    - in diffusion models 46
  - reactive muffler, model of 35
  - realdot 295
  - reattachment length 155
  - reference coordinates 455
  - reference frames 455, 458, 467
  - reference time derivative 456
  - relative permeability 80
  - relative permittivity 80
  - remanent displacement 81
  - remanent magnetic flux density 81
  - remeshing 467
  - repair tolerance 417
  - resistance, in thin layers 430
  - resistive heating 86, 395
  - restarting solver 388
  - Reynolds number 155
    - for a particle 152
  - RF Module 78
  - Robin boundary condition 248
  - ruling application mode 388
- S**
  - Schrödinger equation 268
  - segregated stationary solver 407
  - sensitivity analysis 292
  - sensitivity variables 292
  - shear strain 205
  - shear stress 205
  - shear-rate dependent viscosity 135
  - similarity transformation 485, 488
  - slit boundary conditions 425
  - smoothing methods for moving meshes
    - 466
  - soap film model 354
  - Solid, Stress-Strain application mode 216
  - solution form 268, 270
    - coefficient 268, 270
    - general 268, 270
    - weak 351, 352
  - solver, restarting 388
  - sound hard boundary 25
  - sound soft boundary 25
  - source term 267
    - for two-variable PDE 255
    - variable for 269
  - spatial coordinates 455
  - spatial frames 455, 458, 467
  - spatial time derivative 456
  - specific heat 169, 170
  - speed of sound 24
  - spherical wave 27
  - stabilization
    - crosswind diffusion 183
    - isotropic diffusion 184
    - streamline diffusion 182
  - stagnation point 155
  - states 344

- static analysis 208
  - steady current 85
  - Stefan-Boltzmann constant 174
  - stiffness damping 215
  - stop condition for solver 468
  - strain 205
    - shear 205
  - strain-displacement relations 205
  - streamline diffusion 145, 501
    - for Navier-Stokes equations 156
  - streamline diffusion stabilization 182
  - streamline upwind Petrov-Galerkin 502
  - stress 205
    - normal 205
    - shear 205
    - tensor 205
  - strong formulation vs. weak formulation 369
  - strongly coupled model 390
  - Structural Mechanics Module 204
  - structural mechanics, application modes for 204
  - subdomain settings, specification of 14
  - subset of variables, solving for 389
  - suffixes, for application mode variables 18, 278
  - SUPG. See streamline upwind Petrov-Galerkin
  - surface charge 82
    - boundary condition for 92
  - surface current 82, 99, 109
    - boundary condition for 99, 109
  - surface emissivity 174
  - susceptibility
    - electric 80
    - magnetic 80
  - symmetry boundary condition 174
- T**
- temperature coefficient 86, 401
  - temperature continuity 176
  - temperature gradient 173
  - temperature, boundary condition for 175
  - temperature-dependent conductivity 86
  - temperature-dependent quantity 400
  - test function 368, 371
    - meta variables for 274
  - test function derivatives, meta variable for 274
  - test operator 275, 354
  - thermal analysis, model examples of 188
  - thermal conductivity 169, 171, 173
    - temperature dependent 400
  - thermal expansion 383
  - thermal insulation 174
  - thermal-structural interaction 383
  - thickness 225
    - in plane strain models 225
    - in plane stress models 220
  - thin boundary layer 425
    - for convection-diffusion 55
    - for diffusion 48
  - thin highly capacitive layer 425
  - thin low permeability gap 425
  - thin structures, modeling using prism and brick meshes 442
  - thin thermally resistive layer 176, 425
  - thin-film resistance 430
  - time derivatives 456
    - boundary conditions with 279
    - meta variables for 274
  - TIME suffix 461
  - time-dependent analysis 208
  - time-dependent coefficient form equation 264
  - time-dependent general form equation 258
  - time-dependent problem 249, 264



- in general form 264
    - in weak form 265
  - time-dependent system 249, 265
  - time-harmonic model 267
  - time-scaling coefficient
    - for diffusion models 47
    - in the heat equation 172
  - topology optimization, example of 330
  - transcendental equations 346
  - transient analysis 208
  - transient solver settings, for Navier-Stokes 150
  - transients, avoiding 11
  - typographical conventions 11
- U**
- ua.lu boundary coupled equation variable 271
  - uau boundary coupled equation variable 271
  - ubeu boundary coupled equation variable 271
  - ucu boundary coupled equation variable 271
  - uf boundary coupled equation variable 272
  - uga boundary coupled equation variable 271
  - ultraweak variational formulation 280
  - UMFPACK direct solver, contiguous memory block for 8
  - una.lu boundary coupled equation variable 272
  - unconstrained displacement 485
  - uncu boundary coupled equation variable 272
  - unga boundary coupled equation variable 272
  - unit-depth model. See Plane Strain application mode
- V**
- Vanka preconditioner/smoothen 150
  - variables
    - boundary coupled equation 271
    - for multiphysics couplings only 391
  - variational calculus 354
  - variational principle 370
  - vector differential operator 248
  - velocity, in fluids 135
  - viscosity
    - dynamic 135
    - shear-rate dependent 135
  - viscous damping 209
  - viscous velocity factor 151
  - volume force field 135, 152
  - volume forces 207
- W**
- wave direction cosines 27
  - wave equation 264, 268
  - wave number 29
  - weak constraints
    - applications for 358
    - in assembly geometries 360
    - in structural mechanics 211
    - limitations when using 361
    - specifying 359
  - weak form 5
    - application modes for 351, 369
    - definition of 368
    - edit fields for specifying 274
    - of time-dependent problem 265
    - theoretical background for 368
    - usage of 349
  - weak form contributions, adding 351
  - weak formulation vs. strong formulation 369
  - weak ideal constraints 366
  - weak non-ideal constraints 366
  - weak solution form 351, 352
  - weak terms

- for implementing point sources 286
  - using 275
- weakly coupled model 390
- Winslow smoothing 466
- Y** Young's modulus 206, 212