COMSOL®
RELEASE NOTES

VERSION 3.2b

COMSOL

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@femlab.nl
www.femlab.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
Lauttasaarentie 52
FIN-00200 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
FEMLAB GmbH
Berliner Str. 4
D-37073 Göttingen
Phone: +49-551-99721-0
Fax: +49-551-99721-29
info@femlab.de
www.femlab.de

Italy
COMSOL S.r.l.
Contrada Santa Croce, 22
25125 Brescia
info.it@comsol.com

Norway
COMSOL AS
Verftsgata 4
NO-7485 Trondheim
Phone: +47 73 84 24 00
Fax: +47 73 84 24 01
info@comsol.no
www.comsol.no

Sweden
COMSOL AB
Tegnérgatan 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.
Studio G8 Shepherds Building
Rockley Road
London W14 0DA
Phone:+44-(0)-20 7348 9000
Fax: +44-(0)-20 7348 9020
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.
1100 Glendon Avenue, 17th Floor
Los Angeles, CA 90024
Phone: +1-310-689-7250
Fax: +1-310-689-7527

COMSOL, Inc.
744 Cowper Street
Palo Alto, CA 94301
Tel: +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 3.2b Release Notes

Version: April 2006 COMSOL 3.2b

CONTENTS

Chapter 1: COMSOL 3.2b Release Notes

Chapter 5: MEMS Module 3.2b Model Library Update

Chapter 6: Chemical Engineering Module 3.2b Model Library Update

Chapter 7: Trusses in Structural Mechanics Module 3.2b

Chapter 8: User-Defined Classes in COMSOL Script 1.0b

1

COMSOL 3.2b Release Notes

These Release Notes provide information about new features and changes in the COMSOL 3.2b update. They are intended for users who receive COMSOL 3.2b as part of the COMSOL software maintenance program as well as for new users of the COMSOL 3.2b software products. These notes are a complement to the COMSOL 3.2 printed and online documentation. If you receive the COMSOL 3.2b CDs, this document is available in the root directory of CD 1.

Note: The PDF version of these Release Notes contains supplementary material that do not appear in the printed version.

Introduction

Welcome to COMSOL 3.2b! These *Release Notes* bring you up to date on the new features in this software update.

Obtaining COMSOL 3.2b

If you own a COMSOL license and are an active subscriber to the COMSOL maintenance program, you can download the COMSOL 3.2b software update from http://www.comsol.com/support/updates/.

If you are not an active subscriber to the COMSOL maintenance program or have any questions regarding the COMSOL 3.2b update, contact your COMSOL sales representative. For contact information on COMSOL's worldwide offices and representatives, see www.comsol.com/contact/.

Installing COMSOL 3.2b

If you download the COMSOL 3.2b update from the COMSOL web site, follow the directions provided on the web pages for the software update.

For installations of COMSOL 3.2b using the product CDs, you can find complete information about system requirements and installation procedures in the *COMSOL Installation Guide*.

If you change the license or do a new installation using a Named Single User License (NSL), the installer prompts you for the username to bind to the license and then creates the required options file for you.

System Requirements for 64-bit Windows Versions

- Windows XP Professional x64 Edition
- A PC with one of these processors: AMD Opteron, AMD Athlon 64, Pentium 4 with EM64T, or Xeon with EM64T
- A graphics card with at least 32 MB of memory
- At least 256 MB of system memory

The COMSOL 3.2b Documentation Set

These *Release Notes* are part of the documentation set for COMSOL 3.2b and are available for download from the COMSOL web site as part of the COMSOL 3.2b update.

The COMSOL Reaction Engineering Lab comes with the *COMSOL Reaction Engineering Lab User's Guide*, a printed document with comprehensive information about the software and its applications within chemical reaction engineering.

All other documentation is identical to the COMSOL 3.2 documentation.

COMSOL 3.2b Release Highlights

The COMSOL 3.2b release includes the following new features in the COMSOL software products:

- 64-bit Windows versions of the COMSOL software products for Windows XP Professional x64 Edition.
- A new software package for chemical reaction engineering, the COMSOL Reaction Engineering Lab. It provides full support for the modeling, simulation, and visualization of chemical reactions, import and visualization of experimental data, and provides a seamless interface to the Chemical Engineering Module.
- The ability to use a mesh or a deformed mesh to define a model geometry.
- Remeshing capabilities in COMSOL Multiphysics for improved modeling of moving boundaries and parameterized geometries.
- Support for both reference frames and spatial frames in the dialog box for integration coupling variables. This feature applies to models that include Moving Mesh (ALE) or Parameterized Geometry application modes.
- Improved rendering and graphics in COMSOL Multiphysics and COMSOL Script plot windows. These features include contour labels as well as support for Greek letters, mathematical symbols, and Unicode characters in plot titles, axis labels, legends, and other similar plot elements.
- Improved image export.
- Global expression variables for defining expressions that are valid across all geometries as well as in ODEs and algebraic equations in COMSOL Multiphysics.

- New features in the Electromagnetics Module:
  - Higher-order vector elements
  - Vector elements available for all mesh types
  - Transient analysis in 3D models using the application mode for induction currents using quasi-statics with a magnetic vector formulation
  - Predefined settings and explicit gauge fixing for effective use of the multigrid solvers in magnetostatics and quasi-statics
  - Import of SPICE netlists via COMSOL Script
- New truss/cable elements in the Structural Mechanics Module.
- New predefined multiphysics couplings for flow with species transport in the MEMS Module.
- Support for user-defined objects in COMSOL Script as well as many new functions for image import, polynomials, graphics, operating system commands, sound, and more.
- Support for MATLAB R2006a on Windows, Macintosh, and Solaris. For Linux, a compatibility update is available from www.comsol.com/support/updates.
- Several new and updated models.
- Many small improvements and corrections.

The COMSOL Reaction Engineering Lab

The COMSOL Reaction Engineering Lab is a new software product for the modeling and simulation of chemical systems. The Reaction Engineering Lab automatically generates the chemical kinetics of a reaction process based on reaction formulas you enter. You can add and remove reaction steps or modify rate equations to explore how they affect a reactor's performance. The updated version 1.1 that ships with the COMSOL 3.2b release features support for import of experimental data from files, automatic simulation of the reactions until steady state, and many other improvements.

SEAMLESS INTERFACE TO THE CHEMICAL ENGINEERING MODULE

You can extend any model in the COMSOL Reaction Engineering Lab to include a real-world geometry and multiphysics couplings. It is easy to export the mass balances, heat balances, and momentum balances into corresponding application modes in the Chemical Engineering Module.

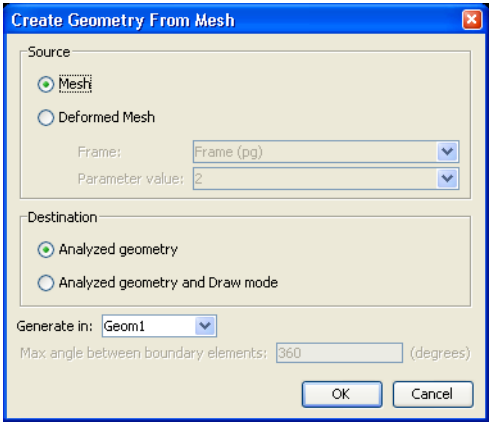
For more information about the COMSOL Reaction Engineering Lab, visit www.comsol.com/products/reaction or contact your local COMSOL representative.

New Features in COMSOL Multiphysics

The following section describes new and improved features in COMSOL Multiphysics.

Creating a Geometry from a Mesh

When you import a mesh there is no corresponding geometry. A similar situation occurs when you have created a deformed mesh and want to remesh using the deformed geometry. COMSOL Multiphysics 3.2b provides the ability to create a geometry from such meshes. This a*nalyzed geometry* is the geometry that you use for modeling. You can also use the created geometry in Draw mode. To generate a geometry from a mesh, choose **Create Geometry from Mesh** from the **Mesh** menu.



In the **Source** area, select if you want to create a geometry from a mesh or if you want to create a geometry from a deformed mesh, which is retrieved from solving a parameterized geometry or a moving mesh (ALE) problem. If you want to create a geometry from a deformed mesh, you also need to specify which frame and solution to use.

In the **Destination** area, select the **Analyzed geometry** option to create only an analyzed geometry, or select the **Analyzed geometry and Draw mode** option to create both an analyzed geometry and a Draw mode object from the mesh. If you select **Deformed mesh** as the source, these options also generate a new mesh, and the destination options are **Mesh and analyzed geometry** and **Mesh, analyzed geometry, and Draw mode**.

You can select from the **Generate in** list in which geometry the software places the generated analyzed geometry. The default destination is the current geometry, but you can also choose from any other geometries in the model or create a new geometry with the generated analyzed geometry.

In the **Max angle between boundary elements** edit field you specify the maximum allowed angle between two boundary elements as part of the same face. This parameter is available only if the mesh has no parameterization.

Note: When you generate an analyzed geometry from a 3D mesh and the **Max angle between boundary elements** parameter is available, the generation may introduce new boundaries.

The equivalent command-line function is `mesh2geom`. Type

```
help mesh2geom
```

for information about the syntax and an example.

See "Remeshing Explained With an Example" on page 7 for an example where you create a new geometry from a deformed mesh.

Frame Selection for Integration Coupling Variables

When working with integration coupling variables and the model includes multiple frames, you can select the appropriate frame from a list on the **Source** tab. The frame determines the volume measure to use when computing the integral.

Remeshing for Moving Mesh and Parameterized Geometries

When you work with the Moving Mesh or Parameterized Geometry application modes, the mesh quality becomes poor when the deformations are large. This section describes how to remesh the geometry after a part of the solution steps and how to continue solving.

SMOOTHING METHODS AND COMPATIBILITY WITH 3.2 MODELS

The Moving Mesh application mode has two smoothing methods, Laplace smoothing and Winslow smoothing, which are responsible for creating a smooth deformed mesh. We have reimplemented Winslow smoothing to work with the remeshing. This new implementation uses mesh positions instead of mesh displacements as degrees of

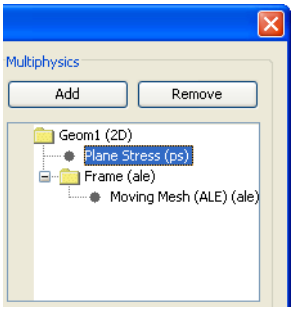
freedom. For backward compatibility, 3.2 models that use Winslow smoothing retain the old implementation. You can change the choice of smoothing method in the **Application Mode Properties** dialog box.

The Parameterized Geometry application mode also includes both smoothing methods.

REMESHING EXPLAINED WITH AN EXAMPLE

The 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 dimensions** list.
2. In the **COMSOL Multiphysics** folder, select **Deformed Mesh>Moving Mesh (ALE)> Transient analysis**.
3. Click the **Multiphysics** button and then 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. Click **OK** to close the dialog box.

2 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** tab and select boundaries 4, 6, and 8. Enter a force that increases with time by setting F_y to `-1e6*t`. Click **OK** to close the dialog box.

3 Select the **Moving Mesh (ALE)** application mode from the **Multiphysics** menu.

4 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`. Click **OK** to close the dialog box.

5 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.

6 Select boundaries 1, 2, 5, 7, and 9, and select both the **dx** and **dy** check boxes to prevent these boundaries from moving. Click **OK** to close the dialog box.

Note that 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** in the **Allow remeshing** list and 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, you must go back, activate it, and re-solve the model.

Mesh Generation

1 Initialize the mesh.

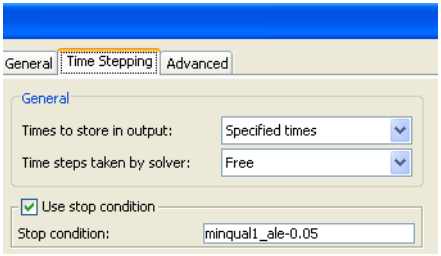
2 Refine it once.

Computing the Solution

In order to make the solver stop when the mesh quality becomes poor, you must 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** tab and select the **Use stop condition** check box.



The solver halts when the expression in the **Stop condition** edit field becomes negative. The default value for the stop condition is `minqual1_ale-0.05`. The Moving Mesh application mode defines the variable `minqual1_ale`, which represents the minimum quality of the deformed mesh. The variable name begins with `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. The name of the 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 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 dictates that the solver should 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`.

3 Click **OK** to close the **Solver Parameters** dialog box and then click the **Solve** button on the Main toolbar.

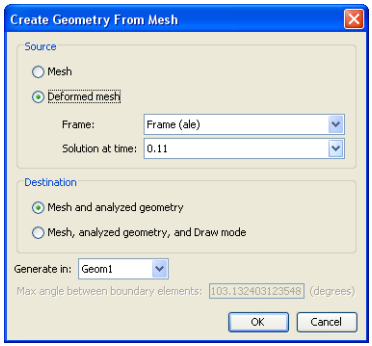
The solver stops somewhere around $t = 0.12$.

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 and then **OK**.



This creates a new geometry from the deformed mesh at the last time step.

Mesh Generation

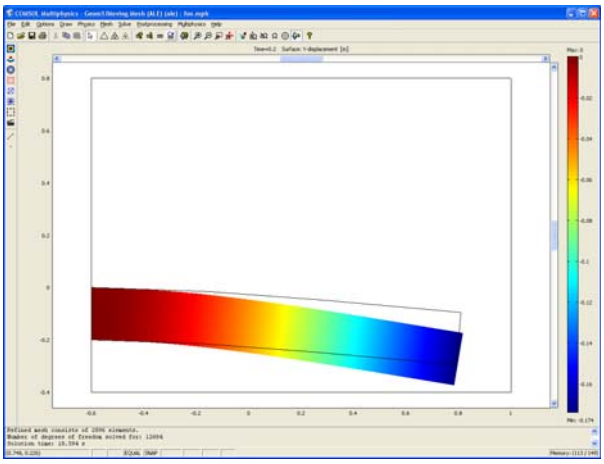
Using buttons on the Main toolbar, initialize the mesh and refine it once. This creates a new mesh with higher quality for the deformed geometry.

Computing the Solution

1 Open the **Solve>Solver Parameters** dialog box and on the **General** tab 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 explicitly depends on the time.

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** toolbar button.

The solver continues solving and stops when the mesh quality again becomes poor. 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 new ability of the `postmovie` command to postprocess several FEM structures (see “Animation Using Multiple FEM Structures” on page 23).



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 22). Using the current Draw mode objects restores the original geometry.

FRAMES WHEN USING REMESHING

When you set the application mode property **Allow remeshing** to **On**, COMSOL Multiphysics adds an extra frame. The three frames are now:

- The *spatial frame*, usually denoted by `ale` 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 `Xm` and `Ym` by default.

LINEAR VS. QUADRATIC ELEMENTS

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 and Parametrized Geometry application mode. 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.

Geometry Repair When Creating Composite Objects

You can now supply a repair tolerance when creating composite objects. The software uses this tolerance to remove small entities and to heal gaps in the generated object. This can greatly improve the handling of imported CAD assemblies. To use this feature, open the **Create Composite Object** dialog box, select the **Repair** check box, then enter a value in the **Repair tolerance** edit field.

Mesh Optimization in 2D

Mesh optimization is now also available in 2D for unstructured triangular meshes. The **Optimize quality** check box on the **Global** tab in the **Mesh Parameters** dialog box controls if COMSOL Multiphysics carries out quality optimization during mesh initialization. The default is to use optimizing of the mesh quality by smoothing the mesh. The smoother that the mesh generator uses when creating meshes improves the mesh quality by jiggling the mesh vertices, that is, by moving each mesh vertex toward the centerpoint of the surrounding mesh vertices and by performing edge swapping.

In COMSOL Script or MATLAB, you can use the corresponding function `meshsmooth` to select among different smoothing techniques and control the smoothing parameters.

Global Expression Variables

Global expression variables are a new type of expression variables. Global expressions are available globally in the model across all geometries (including a 0D "ODE geometry" if you have added extra states or other degrees of freedom in the **ODE Settings** dialog box). They are otherwise similar to scalar expressions variables, which are available only in the current geometry.

To add or edit a global expression variable, go to the **Options** menu, point to **Expressions**, and then click **Global Expressions**.

Function for Numbering of Elements, Nodes, and DOFs

The new function `xmeshinfo` (available in COMSOL Script and MATLAB when you use COMSOL Multiphysics) makes it possible to view information about the extended mesh. Use it to get information about node numbers, node coordinates, degree-of-freedom (DOF) numbers, element numbers, the number of DOFs, and more. The output from `xmeshinfo` contains information that conforms to the data structures in the mesh. Type

```
help xmeshinfo
```

for full information about available properties.

NUMBERING CONVENTIONS

The numbering provided by `xmeshinfo` corresponds to the numbering in the mesh data structure (see `femmesh`). The extended mesh uses a different numbering internally. All numbering is 1-based.

- **Elements**. For each mesh element type, `xmeshinfo` uses the element numbering in `femmesh`.
- **Node points**. The node points in `femmesh` have the same numbers in the extended mesh. Additional node points have higher numbers (these are arbitrarily ordered).
- **Local node points**. The numbering of local node points within a mesh element is different from the numbering in `femmesh`. However, both functions use the same definition of the local coordinate system. In the extended mesh, the local node points are in lexicographical order of their local coordinates. In `femmesh`, the mesh

vertices come first, in lexicographical order, and then come the other node points in lexicographical order (the latter are only present for a second-order mesh).

- **DOFs**. By default, the DOF number is the index in the complete set of degrees of freedom of the model. If you provide a value for the property `Solcomp`, the DOF number is the index in the set of DOFs solved for. If you provide a value for the property `Null`, it is assumed that the Eliminate constraint-handling method is used, and the DOF number is the index in the set of unconstrained DOFs. This assumes a simple form of the constraints where each constraint constrains only one DOF. In other words, each column of the `Null` matrix has a single nonzero element. If `Null` does not have this form, an error message appears. The `Null` matrix is an output from the solvers (see `femlin`).

EXAMPLE OF USING XMESHINFO

Assume that `fem.mesh` is an imported NASTRAN mesh with second-order tetrahedral mesh elements, where node point numbering starts at 1. Use second-order Lagrange elements in the COMSOL Multiphysics model:

```
m = meshimport('nastrandemo1.nas');
fem.mesh = m{1};
fem.dim = 'u';
fem.shape = 2;
fem.equ.c = 1;
fem.bnd.h = 1;
fem.xmesh = meshextend(fem);
```

To get the DOF number corresponding to node point number 22 in the NASTRAN mesh, type

```
nodes = xmeshinfo(fem,'out','nodes');
nodes.dofs(1,22)
```

Compute an eliminated stiffness matrix and a null-space matrix with the line

```
[Kc,Null]=femlin(fem,'out',{'Kc' 'Null'});
```

To find the node point number corresponding to column 30 of `Kc` and its coordinates, type

```
dofs = xmeshinfo(fem,'out','dofs','null',Null);
n = dofs.nodes(30)
dofs.coords(:,30) % alternatively: nodes.coords(:,n)
```

To find the six DOF numbers in tetrahedron element 10 of the mesh, type

```
elements = xmeshinfo(fem,'out','elements','meshtype','tet2');
elements.dofs(:,10)
```

To find the total number of DOFs on the boundary, type

```
xmeshinfo(fem,'out','ndofs','edim',2)
```

The output from this function call reveals that there are 33,702 DOFs on the boundary.

Stop Condition for the Time-Dependent and Parametric Solvers

On the **Time Stepping** and **Parametric** tabs in the **Solver Parameters** dialog box there is now a **Use stop condition** check box. If you select this check box, the solver stops before the expression in the **Stop condition** edit field becomes negative (the solver does not try to resolve the exact time for the zero crossing). The expression must be a scalar value that has no variation in space, typically a coupling variable with a global destination. When using a deformed mesh with remeshing, the Moving Mesh application mode provides a default expression that is the minimum quality of the deformed mesh minus 0.05 (see "Computing the Solution" on page 8).

Stop if Error Due To Undefined Operations

By default, the solver stops with an error message when it encounters an undefined mathematical operation in an expression that appears in the model settings, for instance, division by zero or square root of a negative number. To change this behavior, go to the **Advanced** tab of the **Solver Parameters** dialog box and clear the **Stop if error due to undefined operation** check box. Then the solver treats the result of the operation as `Inf` (infinity) or `NaN` (not a number). This feature can be useful in a nonlinear problem where the steps in the iterative solution process lead to variable values for which an expression is undefined. The solver then reduces the step size in the Newton iteration when it encounters `Inf` or `NaN` so that it can find a solution. The corresponding property for the command-line solver functions is `matherr on/off`.

Plotting Global Expressions

To visualize globally defined variables such as solutions to ODEs, use the **Global Variables Plot** dialog box.

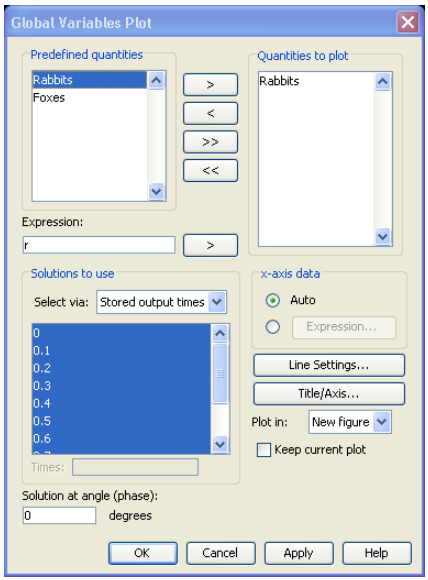


Figure 1-1: The Global Variables Plot dialog box.

The **Predefined quantities** list includes the names of all ODE variables (defined in the **ODE Settings** dialog box) and global expressions (defined in the **Global Expressions** dialog box). To plot one or more of these, select them and click the **>** button to add them to the **Quantities to plot** list. When you select a name in the **Predefined quantities** list, the software displays its variable name in the **Expression** edit field below the list. You can edit this name and add the edited quantity using the **>** button to the right of the text field. To remove one or more expressions from the **Quantities to plot** list, select them and click the **<** button. You can use Shift-click and Control-click in both lists.

In the **Solutions to use** area, select which time steps to include in the plot. By choosing **Interpolated times** in the **Select via** list, you can enter any vector of times in the **Times** edit field, such as `linspace(0,1,100)` or `0:0.02:1`.

Control the quantity to plot on the *x*-axis in the **x-axis data** area. If **Auto** is selected, the *x*-axis corresponds to time.

The other settings in this dialog box are similar to those in the **Cross-Section Plot Parameters** and **Domain Plot Parameters** dialog boxes (see Chapter 7, "Postprocessing and Visualization," in the *COMSOL Multiphysics User's Guide*)

To plot global variables in a script or at the command line, use the corresponding function `postglobalplot`.

Contour Labels

You can now use labels that display the values of contours (isolevels). To include contour labels in a contour plot, open the **Plot Parameters** dialog box, click the **Contour** tab, and select the **Labels** check box in the **Contour levels** area.

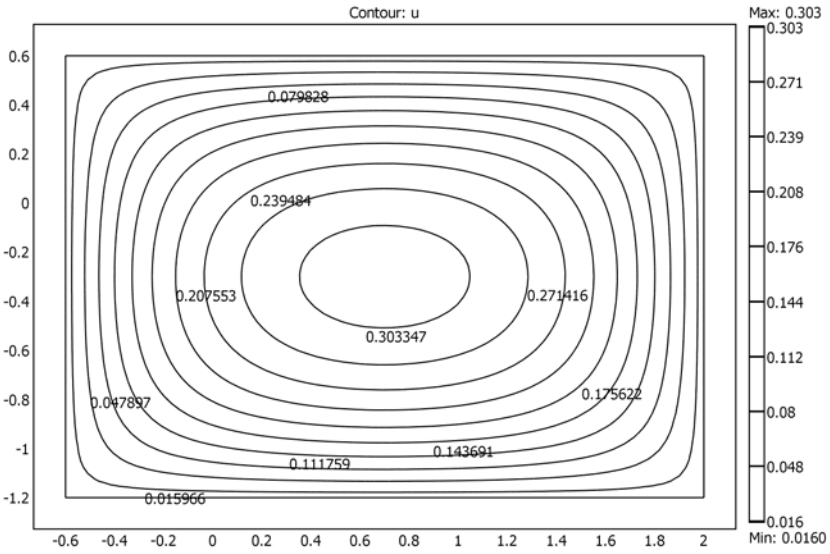


Figure 1-2: Contour labels in a plot using ten contour levels.

Improved Streamline Plots and Particle Tracing

COMSOL Multiphysics 3.2b features reorganized **Streamline** and **Particle Tracing** tabs in the **Plot Parameters** dialog box. The **Streamline** tab is only for stationary fields, and the **Particle Tracing** tab now includes an option to use either massless particles or particles with mass. Select which to use in the **Particle type** list and then click the **With Mass** or **Massless** tab to specify the equation of motion.

In addition to global coordinates, you can now use boundary coordinates to select start points on boundaries for the particles. On the **Particle Tracing** tab, click the option

button to the left of the **Boundary Coordinates** button; then click that button to open the **Boundary Coordinates** dialog box. Select the boundaries where you want your starting points. Click the **Number of points** button to use a number of equidistant points. Click the **Vector with boundary parameters** button to enter a vector of values from 0 to 1 to describe the start points. From the command line or in a script, use the `postcoord` function, which is the corresponding command-line function for computing the start points. The general plot function `postplot` can also call `postcoord` to compute the start points. Use the `postpart` property in `postplot` to pass a cell array with property/value pairs to `postcoord`.

Improved Scripting Support in COMSOL Script

The functions in the following table are now available in COMSOL Script:

FUNCTION	DESCRIPTION
`flcontour2mesh`	Create boundary mesh from contour data
`flim2curve`	Create curve2 object from image data
`flmesh2spline`	Create spline curves from the mesh
`geomspline`	Spline interpolation
`helix2`	Create face helix in 3D
`helix3`	Create solid helix in 3D
`loft`	Loft 2D geometry sections to 3D geometry object

These additions mean that code using these functions work in COMSOL Script as well as MATLAB, and notes in the documentation that state you must run COMSOL Multiphysics with MATLAB no longer apply. For more information about these functions, use the command-line help in COMSOL Script or see the *COMSOL Multiphysics Command Reference*.

Texts Including HTML Formatting, Math Symbols, and More

Text you add to plots using COMSOL Script can now include HTML tags, Greek letters, and mathematical symbols. You can also display other characters using the corresponding Unicode numbers. These options are also available for titles of postprocessing plots in COMSOL Multiphysics. For more information about these options, see “Support for Formatting and Symbols in Texts” on page 34.

Please note several other smaller improvements:

- Using custom fonts and custom font sizes now works in colorbars and legends
- *x*-axis labels can now print to the right or left of the axis panel
- Major and minor tick marks are available when using log scales

Improved Image Export

Using the new and improved **Export Image** dialog box, you can now control font sizes, line widths, and what to include in an exported image. To speed up the image-generation process, a preview feature and image rendering information are available.

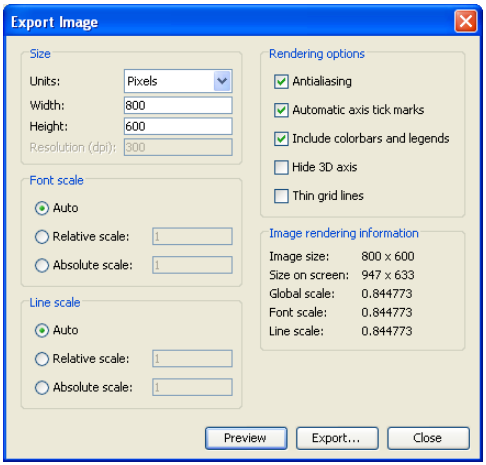


Figure 1-3: The Export Image dialog box.

SETTINGS IN THE EXPORT IMAGE DIALOG BOX

Scaling

Settings in the **Font scale** and **Line scale** areas affect the scaling between the plot's size on the screen and the size in the image (size = number of pixels):

- Click **Auto** to use the global scale (you see its value in the **Image rendering information** area; also see "The Global Scale" on page 20) if you specify the size in pixels (the software scales text, lines, and other graphics equally). If you specify the size in centimeters or inches, the automatic scale is based on the resolution that you select. The font size and line width you specify when creating the plot are preserved if you

export an image using a certain resolution in dpi (dots per inch) and import it to a document as an image using the same dpi resolution (a text with a certain size in the plot looks like a text with the same size in the document).

- Click **Relative scale** to use a total font scale that is the automatic scale times the relative scale you specify.
- Click **Absolute scale** to use a total font scale that is equal to the absolute scale you specify.

RENDERING OPTIONS

The following settings are available in the **Rendering options** area:

Antialiasing

Select the **Antialiasing** check box to reduce stairstep-like lines (jaggies) and make lines and edges look smooth.

Automatic Tick Marks

Select the **Automatic tick marks** check box to take advantage of a new feature that can hide axis tick marks if they overlap. Clear this check box if you want make sure that the image has the same axis tick marks as you see on the screen.

Hiding the 3D Axis

Select the **Hide 3D axis** check box to exclude the grid and coordinate system from the image.

Rendering Thin Grid Lines

Select the **Thin grid lines** check box to render thin grid lines compared to other lines in the image. Use this option if you think that the grid is too dominating in the image.

THE GLOBAL SCALE

The global scale is the scale between the size of the plot on screen and the size of the image (size = number of pixels).

Explicit View Settings

You can now can accurately control camera parameters such as the position, target, and view angle. To do so, open the **Visualization/Selection** dialog box from the **Options** menu and click the **Camera** tab.

The view settings for the camera parameters are available only in 3D.

The corresponding command-line functions (`campos`, `camtarget`, `camva`, `camup`, and `camproj`) are available in COMSOL Script.

Antialiasing and Line Width as Preference Settings

It is now possible to use antialiasing (smoothing graphics) when rendering in COMSOL Multiphysics. You can also change the line width.

To specify antialiasing and the line width, open the **Preferences** dialog box. Click the **Visualization** tab to access these properties.

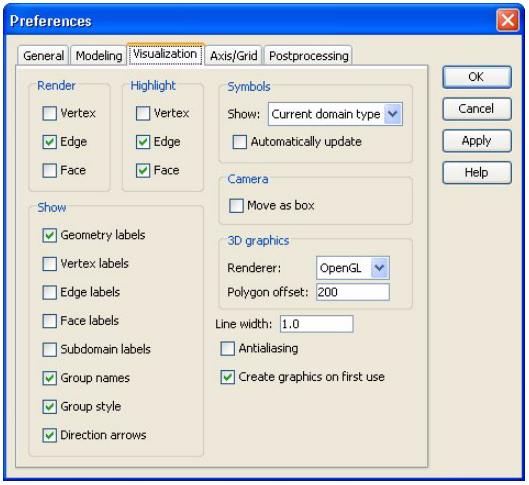


Figure 1-4: The Visualization tab in the Preferences dialog box.

ANTIALIASING

To use antialiasing, select the **Antialiasing** check box. Antialiasing reduces the rendering speed, so the default setting is to not use antialiasing.

LINE WIDTH

Specify the line width as a scalar number in the **Line width** edit field. Figure 1-5 shows the effect of increasing the line width.

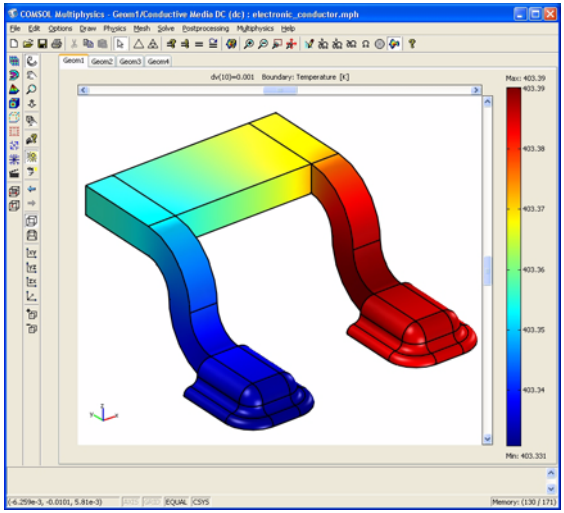
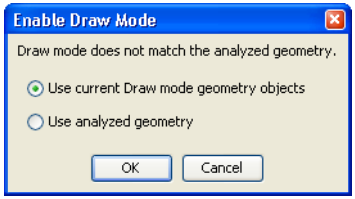


Figure 1-5: The effect of increasing line width.

Entering Draw Mode

In certain situations, the Draw mode contents do not match the current analyzed geometry. This happens, for instance, when you replace the geometry and the mesh with a deformed mesh so that the Draw mode still contains the original geometry while the analyzed geometry is the deformed geometry.

If you try to enter Draw mode when it does not match the analyzed geometry, the software displays a dialog box where you can choose between keeping the geometry objects in Draw mode or replacing them with the analyzed geometry. Then click **OK** to enter Draw mode.



If there is no analyzed geometry, as when working with an imported mesh, the option to use the analyzed geometry is not available. In that case you can use the **Create**

Geometry From Mesh dialog box in the **Mesh** menu to create an analyzed geometry and a Draw mode object from the imported mesh.

Note: If you open Draw mode using the **Use current Draw mode geometry objects** option, this action invalidates the old geometry, mesh, and solution.

New Command-Line Features for Postprocessing

NEW FUNCTION: POSTGLOBALEVAL FOR EVALUATING ODE VARIABLES

The new function `postglobaleval` evaluates the ODE variables in `fem.ode.dim` and puts the results in a structure with the fields `x`, `y`, and `legend`. Here `x` is an array that contains the time steps, and `y` is a matrix that contains the values of all ODE variables where each column corresponds to an ODE variable. Further, `legend` is a cell array of legend strings. You can use `postglobaleval` to create data for plotting the solution to ODEs using the following code snippet if the FEM structure `fem` contains the solution to the ODEs:

```
data = postglobaleval(fem);
plot(data.x, data.y)
legend(data.legend)
```

IMPROVED SELECTION OF EVALUATION POINTS IN POSTEVAL

Use the new properties `Spoint`, `Bpoint`, and `Prpoint` in `posteval` to specify arbitrary local element evaluation points for simplex elements (triangular, tetrahedral, and edge elements), quadrilateral/block elements, and prism elements, respectively. If you specify any of these properties, the fields `t` and `q` in the output structure are empty, and the property `Cont` is neglected. To evaluate in Gauss points, use the new function `postgp` which returns Gauss points and Gauss point weights. You can use the output from `postgp` as input to the properties `Spoint`, `Bpoint`, and `Prpoint` in `posteval`.

The output structure from `posteval` now also includes the field `elind` with indices to the mesh element for each point.

ANIMATION USING MULTIPLE FEM STRUCTURES

The function `postmovie` can now take a cell array of FEM structures as input. This makes it possible to make animations of models using manual remeshing. For example, when you use the property `stopcond` in a solver in the user interface and then export

the FEM structure after each stop with the names `fem1`, `fem2`, and so on, you can make an animation of the entire model using the following command:

```
postmovie({fem1,fem2,fem3,...}, ...)
```

Updated and Corrected Models

Updated models are available with the COMSOL Multiphysics 3.2b update on the COMSOL web site at http://www.comsol.com/support/updates/ and on the COMSOL 3.2b CDs.

ELECTRONIC CONDUCTOR

This introductory model, which appears in the *COMSOL Multiphysics Quick Start and Quick Reference*, includes a geometry file, `new_soldering_geom.mphbin`, which you can use if you want to skip the geometry modeling steps. This file was missing in COMSOL 3.2 but is now available for download from the COMSOL web site.

Documentation Erratum

Step 21 on page 62 in the *COMSOL Multiphysics Quick Start and Quick Reference* should say:

- In the **Name** edit field enter `air`.

EIGENMODES OF A ROOM

The eigenvalue formulation in the Acoustics application mode has changed from first-order eigenvalues to second-order eigenvalues. To get the same eigenfrequencies in the analysis, use another setting in the **Search for eigenvalues around** edit field. Replace Step 2 in "Computing the Solution" on page 26 in the *COMSOL Multiphysics Model Library* with the following:

- The frequency 100 Hz corresponds to $-i\omega = -i2\pi \cdot 100 = -i628$. Enter `15` in the **Desired number of eigenvalues** edit field and `-i1e2` in the **Search for eigenvalues around** edit field.

SHALLOW WATER EQUATIONS

The updated Shallow Water Equations model uses artificial stabilization and tighter tolerances for the time stepping. Full model documentation is available in the PDF version of these Release Notes at http://www.comsol.com/support/updates/ and on the COMSOL 3.2b CD 1.

SLOSHING TANK

The updated Sloshing Tank model uses correct constraints for the free surface. Full model documentation is available in the PDF version of these Release Notes at http://www.comsol.com/support/updates/ and on the COMSOL 3.2b CD 1.

PACEMAKER ELECTRODE

The following changes to the step-by-step instructions correct the geometry modeling for the Pacemaker Electrode model.

Geometry Modeling

- After Step 1 on page 67 in the *COMSOL Multiphysics Model Library*, add the following step:

  Click the **y-z** button and then click **OK**.

- In Step 7 on page 67, change the radius entry to `1e-3`.
- In Step 3 for the hooks on page 68, click the Chamfer button to create a chamfer instead of a fillet. Type `2e-4` in the **Distance** edit field.

New Features in the Electromagnetics Module

Higher-Order Vector Elements

In addition to linear vector elements, the Electromagnetics Module now makes 2nd and 3rd-order vector elements available. These higher-order vector elements provide increased accuracy in electromagnetic wave simulations. You can select from **Vector - Linear**, **Vector - Quadratic**, and **Vector - Cubic** in the lists of elements for all application modes that use vector elements.

For an example model that uses higher-order vector elements, see "New Model—Inductance of a Power Inductor" on page 28.

Vector Elements for All Mesh Element Types

You can now use the linear and higher-order vector elements with all mesh element types. That is, the vector elements work with unstructured meshes as well as mapped, extruded, and revolved meshes.

Multigrid Support in Electromagnetic Wave Models

The multigrid solver now supports the use of nested meshes and different element orders for models that include vector elements.

Multigrid Support for AV Quasi-Statics and Magnetostatics Models

You can now use the multigrid solver for AV quasi-statics and magnetostatics (vector potential formulation) with explicit gauge fixing. Predefined settings make it easy to take advantage of the accelerated performance that the multigrid solver provides.

Transient Quasi-Statics, Vector Potential—Induction Currents

The 3D application mode for induction currents using quasi-statics with a magnetic vector formulation now supports transient analysis in addition to time-harmonic analysis. To access this option from the **New** tab in the **Model Navigator**, select **3D** in the

Space dimension list and then in the list of application modes browse to **Electromagnetics Module>Quasi-Statics, Magnetic>Induction Currents>Transient analysis**.

Import of SPICE Netlists via COMSOL Script

With the `spiceimport` script you can import a SPICE netlist into an ODE structure representing the circuit in the file. The script assumes that netlists are in the original Berkeley netlist syntax for devices and models (http://www.eecs.berkeley.edu/). Most circuit simulators can export to this format.

Not all device models are implemented, and those implemented do not support the full range of parameters available. It might therefore be necessary to simplify the netlist prior to the import. Currently `spiceimport` supports the devices in Table 1-1 below with the listed limitations.

TABLE 1-1: SUMMARY OF SUPPORTED DEVICES

SYMBOL	DESCRIPTION	LIMITATION
R	Resistor	No temperature dependence
C	Capacitor	No voltage and temperature dependence
L	Inductor	No current and temperature dependence
V	Independent voltage source	Support constant sources, pulse sources, and sine sources. Variable names can be used to implement arbitrary expressions by adding them later in Global expressions
I	Independent current source	See above
E	Voltage-controlled voltage source	Gain controlled source
F	Current-controlled current source	See E device
G	Voltage-controlled current source	See E device
H	Current-controlled voltage source	See E device
D	Diode	No temperature dependence
Q	Bipolar transistor	Implements parts of the Gummel-Poon transistor model
M	MOS transistor	Implements the MOS transistor model as defined by Shichman and Hodges

New Model—Inductance of a Power Inductor

Power inductors are a central part of many low-frequency power applications. This model shows an inductance calculation on a large 3D geometry using higher-order vector elements and memory-efficient iterative solver settings.

This model is included on the COMSOL 3.2b CDs and is available for download at http://www.comsol.com/support/updates/. Full model documentation is available in the PDF version of these Release Notes at http://www.comsol.com/support/updates/ and on the COMSOL 3.2b CD 1.

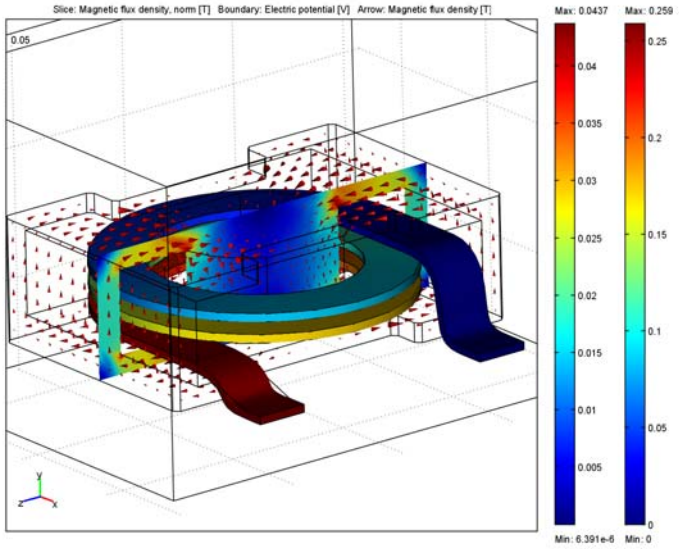


Figure 1-6: This plot of a power inductor shows the potential on the coil, the magnitude of the flux density inside the ferrite core, and the direction of the flux as arrows.

New Features in the MEMS Module

Predefined Multiphysics Couplings for Flow with Species Transport

A suite of predefined multiphysics couplings for flow with species transport is now available in the MEMS Module through its list of application modes in the **Microfluidics** folder. The **Flow with Species Transport** folder includes the following predefined combinations of microfluidic application modes with Convection and Diffusion application modes:

- Non-Isothermal Flow plus Convection and Diffusion
- Incompressible Navier-Stokes plus Convection and Diffusion
- Non-Isothermal Stokes Flow plus Convection and Diffusion
- Stokes Flow plus Convection and Diffusion
- General Laminar Flow plus Convection and Diffusion

All these combinations support transient and steady-state analysis in 2D, axisymmetric 2D, and 3D.

Each predefined coupling provides the velocity components from the microfluidic application mode as the default values for the velocity in the convective part of the Convection and Diffusion application mode.

EXAMPLE MODEL—MICROCHANNEL H-CELL

The Microchannel H-Cell is a new model in the MEMS Module Model Library, that makes use of the predefined multiphysics coupling for flow with species transport. The model treats a microchannel H-cell for separation through diffusion.

New Models

The following new models are included on the COMSOL 3.2b CDs and are available for download at http://www.comsol.com/support/updates/. Full model documentation is available in the PDF version of these Release Notes at http://www.comsol.com/support/updates/ and on the COMSOL 3.2b CD 1

ELECTROOSMOTIC BIOCHIP

This example models the electroosmotic flow in a DNA chip. Fluid displacement takes place due to the presence of a charged solution at the walls of the chip's channels, and

the fluid migrates when an electric field is applied. The electroosmotic flow in the model couples the electric field to the Stokes Flow application mode at the boundaries.

THERMOELASTIC DAMPING IN 2D AND 3D

Thermoelastic damping is an important factor when designing MEMS resonators. The cyclic deformation of the resonator creates local temperature variations, which result in thermal expansion of the material. The expansion appears as damping. These models show how to model thermoelastic damping in a MEMS resonator using fully coupled thermal and structural equations in both 2D and 3D.

MICROCHANNEL H-CELL

See "Example Model—Microchannel H-Cell" on page 29.

New Features in the Structural Mechanics Module

The Truss Application Modes

Truss application modes in 2D and 3D are now available in the Structural Mechanics Module. You can use trusses to model lattice works, often combined with beams. This removes the need to use several Euler Beam application modes to model a moment-free connection between beams. Instead you can use beams and trusses. Trusses are available on boundaries in 2D and edges in 3D. A truss is a component capable of withstanding only axial forces. Using the Truss application modes you can solve hanging cable problems as well as more standard truss problems. The degrees of freedom are the global displacements. The application modes use Lagrange shape functions and tangential derivatives and support the following analysis types:

- Static analysis
- Nonlinear static analysis
- Frequency response analysis
- Eigenfrequency analysis
- Quasi-static transient analysis
- Parametric analysis
- Transient analysis
- Linear buckling analysis

The Truss application modes also include these features:

- Coordinate systems on all domain levels
- Large deformations
- Initial stresses and strains
- Thermal coupling
- Possibility to constrain an edge to be straight to solve the classic singular problem for trusses.
- Possibility to model hanging cable problems where the edge is not straight

EXAMPLE MODEL—IN-PLANE TRUSS

The In-Plane Truss example model contains a linear static analysis of a simple 2D pin-jointed truss. The solution to this benchmark problem correlates well with analytical results.

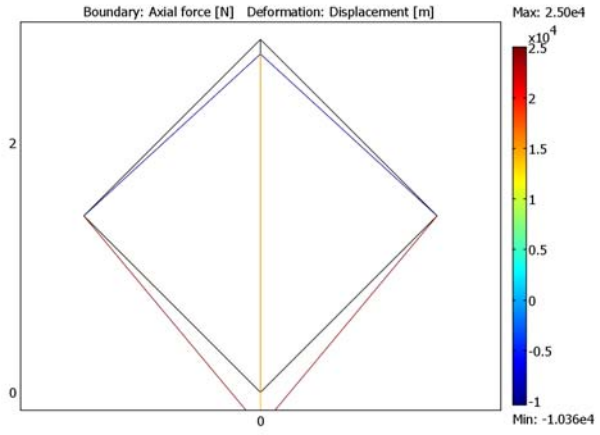


Figure 1-7: Axial forces and deformation for an in-plane truss model.

COMPLETE TRUSS APPLICATION MODES DOCUMENTATION

Complete documentation for the new Truss application modes is available in the PDF version of these Release Notes on the COMSOL 3.2b CD 1 and the COMSOL web site at http://www.comsol.com/support/updates/.

Updated and Corrected Models

The new In-Plane Truss model and updated versions of the following models are available at http://www.comsol.com/support/updates/ and on the COMSOL 3.2b CDs. Additional documentation for some of the models is available in the PDF version of these Release Notes on the COMSOL 3.2b CD 1 and the COMSOL web site at http://www.comsol.com/support/updates/.

ACOUSTIC-STRUCTURE INTERACTION

The eigenvalue formulation in the Acoustics application mode now provides second-order eigenvalues. To match the first-order eigenvalues in the Mindlin Plate application mode, add the following steps before the steps in the section "Subdomain Settings" on page 19 in the *Structural Mechanics Module Model Library*:

Subdomain Settings

1 On the **Physics** menu, point to **Equation System**, and then click **Subdomain Settings**.

2 Copy the expression in the $\mathbf{e_a}$ edit field and paste it into the $\mathbf{d_a}$ edit field.

3 Type `0` in the $\mathbf{e_a}$ edit field and then click **OK**.

BLADDER

This updated version of the Bladder model uses realistic values for the Rayleigh damping and individual tolerance settings for improved accuracy. Make the following changes to the step-by-step descriptions in the *Structural Mechanics Module Model Library*:

Subdomain Settings

Replace Steps 21 and 22 on page 138 with the following steps:

- Click the **Material** tab and select all subdomains to specify the same Rayleigh damping parameters to all subdomains.
- Set $\alpha_{\mathbf{dM}}$ to `1.88` and $\beta_{\mathbf{dK}}$ to `0.016`.
- Click **OK**.
- In the **Physics** menu, select **Properties**. Then select **On** in the **Large Deformation** list.

Computing the Solution

Add the following steps after Step 3 on page 139:

- Type `1e-3` in the **Relative tolerance** edit field.
- Type `u 2e-7 v 3e-6 w 5e-7` in the **Absolute tolerance** edit field to specify individual tolerances for each displacement component.

In addition, we recommend that you run the simulation for only a fraction of the full two seconds (0.2 seconds, for example), because of the size of the model.

COUPLED VIBRATIONS

The model now uses some simplified settings for the density and the normal acceleration.

TRAFFIC TUNNEL

The updated version of this model shows how to use a Mohr-Coulomb nonlinear material model by making the Drucker-Prager criterion identical to the Mohr-Coulomb criterion. This version also uses more realistic material data and a simplified modeling approach with two application modes instead of three.

New Features in COMSOL Script

Improved Plots

You can now use logarithmic scales on both the x-axis and the y-axis.

Support for Formatting and Symbols in Texts

The `text` function can now take formatted strings that include HTML tags, Greek letters, mathematical symbols, and Unicode characters. These formatting options include the strings in plot titles as well as x-axis, y-axis, and z-axis labels (the `title`, `xlabel`, `ylabel`, and `zlabel` functions, respectively).

The `text` function supports the following HTML tags in the text string:

TABLE 1-2: VALID HTML TAGS

HTML TAG	DESCRIPTION
<B> </B>	Enclosed text is rendered using a bold font.
 	Line break.
<CENTER> </CENTER>	Centered text.
<I> </I>	Enclosed text is rendered using an italic font.
<LI>	List item. When the list used is <OL> (an ordered list) the LI element is rendered with a number. When the list used is <UL> (an unordered list) the LI element is rendered with a bullet.
<OL> </OL>	Ordered list (see also: <LI>).
<P> </P>	Paragraph. This tag creates a line break and a space between lines.
<PRE> </PRE>	Enclosed text preserves spaces and line breaks. The text is rendered using a monospaced font.
<STRIKE> </STRIKE>	Enclosed text is rendered with a strike-through appearance.
	Enclosed text is rendered in subscript with the enclosed text slightly lower than the surrounding text.
	Enclosed text is rendered in superscript with the enclosed text slightly higher than the surrounding text.
<TT> </TT>	Enclosed text is rendered using a monospaced font.

TABLE 1-2: VALID HTML TAGS

HTML TAG	DESCRIPTION
<U> </U>	Enclosed text will be underlined.
<UL> </UL>	Unordered list (see also: <LI>).

The `text` function supports the following Greek character tags in the text strings:

TABLE 1-3: VALID GREEK SYMBOL TAGS

TAG	SYMBOL	TAG	SYMBOL
\ALPHA	Α	\alpha	α
\BETA	Β	\beta	β
\GAMMA	Γ	\gamma	γ
\DELTA	Δ	\delta	δ
\EPSILON	Ε	\epsilon	ε
\ZETA	Ζ	\zeta	ζ
\ETA	Η	\eta	η
\THETA	Θ	\theta	θ
\IOTA	Ι	\iota	ι
\KAPPA	Κ	\kappa	κ
\LAMBDA	Λ	\lambda	λ
\MU	Μ	\mu	μ
\NU	Ν	\nu	ν
\XI	Ξ	\xi	ξ
\OMICRON	Ο	\omicron	ο
\PI	Π	\pi	π
\RHO	Ρ	\rho	ρ
\SIGMA	Σ	\sigma	σ
\TAU	Τ	\tau	τ
\UPSILON	Υ	\upsilon	υ
\PHI	Φ	\phi	φ
\CHI	Χ	\chi	χ
\PSI	Ψ	\psi	ψ
\OMEGA	Ω	\omega	ω

The `text` function supports the following mathematical symbol tags in the text string:

TABLE 1-4: VALID MATHEMATICAL SYMBOL TAGS

TAG	SYMBOL	TAG	SYMBOL
\approx	≈	\bullet	•
\lequal	≤	\partial	∂
\gequal	≥	\nabla	∇
\plusmin	±	\sqrt	√
\infinity	∞	\integral	∫

In addition to these Greek and math symbols, you can specify additional characters using Unicode numbers. Visit http://www.unicode.org for more information about Unicode characters.

EXAMPLES OF FORMATTED TEXTS

To plot the following mathematical text in the position (1, 2) in the current axes

$$\sin(2\pi x_i) \approx 0\,,$$

type:

```
text(1,2,'sin(2\pix<SUB>i</SUB>) \approx 0','FontName','Arial',...
'FontSize',16)
```

This example also specifies the font name and size using the optional FontName and FontSize properties.

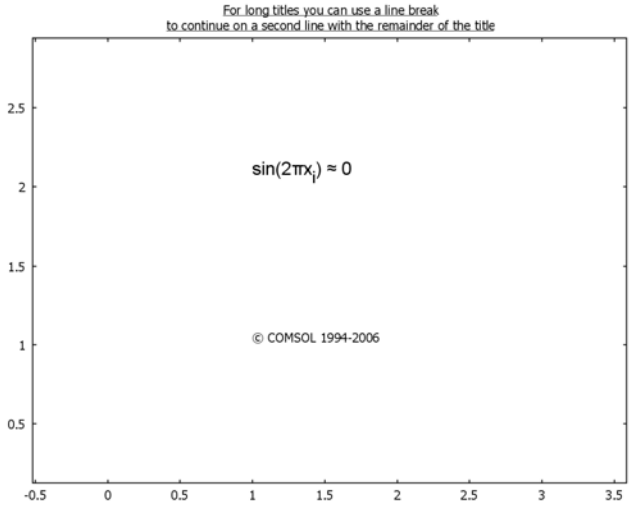
To add a text that includes the copyright symbol, you can use its Unicode:

```
text(1,1,'\u00A9 COMSOL 1994-2006')
```

To add an underlined title to a plot with the text divided into two lines, type:

```
title('<U>For long titles you can use a line break<BR>to continue
on a second line with the remainder of the title</U>')
```

The following figure shows the text strings in this example in a COMSOL Script window.



New Functions

There are a number of new functions in COMSOL Script 3.2b compared to version 3.2. For more information about these functions, type `help` followed by the function name at the COMSOL Script command prompt. The following tables list the new functions divided into the following categories:

- Graphics functions (Table 1-5 on page 38)
- Image functions (Table 1-6 on page 38)
- I/O functions (Table 1-7 on page 38)
- Linear algebra functions (Table 1-8 on page 39)
- Object functions (Table 1-9 on page 39)
- Operating system functions (Table 1-10 on page 39)
- Polynomial and spline functions (Table 1-11 on page 39)
- Sparse matrix functions (Table 1-12 on page 40)
- Specialized mathematical functions (Table 1-13 on page 40)
- Statistics functions (Table 1-14 on page 40)
- Zero-finding function (Table 1-15 on page 41)

GRAPHICS FUNCTIONS

TABLE 1-5: GRAPHICS FUNCTIONS

FUNCTION	DESCRIPTION
bar	Create a bar graph
campos	Control the camera position
camtarget	Control the camera target
camup	Control the camera up vector
camva	Control the camera view angle
clabel	Add labels to a contour plot
contour	Create a contour plot
contour3	Create a 3D contour plot
countourc	Compute a contour data matrix
contours	Compute a contour data matrix
errorbar	Error bar plot
trimesh	Create a mesh plot with triangles
trisurf	Create a surface plot with triangles

IMAGE FUNCTIONS

TABLE 1-6: IMAGE FUNCTIONS

FUNCTION	DESCRIPTION
image	Show an image
imagesc	Show an image using scaled mapping
imread	Read an image from file
imshow	Show an image
imwrite	Write an image to file

I/O FUNCTIONS

TABLE 1-7: I/O FUNCTIONS

FUNCTION	DESCRIPTION
sound	Play sound
soundsc	Play sound (scaled)
wavread	Read a .wav sound file
wavwrite	Write a .wav sound file

LINEAR ALGEBRA FUNCTIONS

TABLE 1-8: LINEAR ALGEBRA FUNCTIONS

FUNCTION	DESCRIPTION
`funm`	Evaluate a matrix function
`logm`	Matrix logarithm
`ordschur`	Reorder Schur factorization

OBJECT FUNCTIONS

TABLE 1-9: OBJECT FUNCTIONS

FUNCTION	DESCRIPTION
`clone`	Create a copy of an instance of a user-defined class
`methods`	Get the methods provided by a user-defined class
`super`	Run superclass constructor
`this`	Get the instance for which an instance method is run

OPERATING SYSTEM FUNCTIONS

TABLE 1-10: OPERATING-SYSTEM FUNCTIONS

FUNCTION	DESCRIPTION
`cputime`	CPU time (in seconds)
`dir`	Get a list of the files in a directory
`dos`	Run a DOS command
`encrypt`	Encrypt M-files
`isdir`	True if a directory exists
`mkdir`	Create a directory
`profile`	Generate profiling information
`rmdir`	Remove a directory
`system`	Run a system command
`unix`	Run a system command

POLYNOMIAL AND SPLINE FUNCTIONS

TABLE 1-11: POLYNOMIAL AND SPLINE FUNCTIONS

FUNCTION	DESCRIPTION
`delaunay`	Delaunay triangulation
`delaunay3`	3D Delaunay triangulation

TABLE 1-11: POLYNOMIAL AND SPLINE FUNCTIONS

FUNCTION	DESCRIPTION
mkpp	Make a piecewise polynomial
pchip	Piecewise cubic Hermite interpolation
ppval	Evaluate a piecewise polynomial
spline	Cubic spline interpolation
unmkpp	Extract details from piecewise polynomial

SPARSE MATRIX FUNCTIONS

TABLE 1-12: SPARSE MATRIX FUNCTIONS

FUNCTION	DESCRIPTION
spdiags	Sparse diagonal or band matrix manipulation
spones	Sparse matrix of ones
sprand	Sparse random matrix with uniformly distributed numbers
sprandn	Sparse random matrix with normally distributed numbers
sprandsym	Symmetric sparse random matrix

SPECIALIZED MATHEMATICAL FUNCTIONS

TABLE 1-13: SPECIALIZED MATHEMATICAL FUNCTIONS

FUNCTION	DESCRIPTION
betainc	Incomplete beta function
gammainc	Incomplete gamma function

STATISTICS FUNCTIONS

TABLE 1-14: STATISTICS FUNCTIONS

FUNCTION	DESCRIPTION
corrcoef	Correlation coefficients
cov	Covariance matrix
subspace	Principal angle between subspaces

ZERO-FINDING FUNCTION

TABLE 1-15: ZERO-FINDING FUNCTION

FUNCTION	DESCRIPTION
`fzero`	Find a zero of a function

Updated and Improved Functions

- The `text` function supports HTML formatting, Greek letters, mathematical symbols, and Unicode characters. See "Support for Formatting and Symbols in Texts" on page 34 for details.
- The `uint8` function returns an unsigned integer (uint8 array). All other functions for converting doubles to integers and unsigned integers (`int*`, `uint*`) are unchanged and return doubles.

User-Defined Classes

Creating custom classes helps you improve the organization of your code. A class is a custom data type that contains data as well as methods operating on objects of the class. Each class is defined by a `csl`-file (short for COMSOL Script language), which defines the fields and methods that the class provides. The object model is inspired by the Java class system, but you do not need any previous experience with Java to use it. For complete documentation on user-defined classes and objects in COMSOL Script, see the PDF version of these Release Notes on the COMSOL 3.2b CD 1 and the COMSOL web site at http://www.comsol.com/support/updates/.

Encrypting M-files

You can protect the contents of custom M-files using the `encrypt` command, which creates encrypted versions of the files in the input arguments. The input files must exist and be valid M-files. For each input file, COMSOL Script creates an MC-file in the current directory (extension `.mc`). When you run an MC-file, it is equivalent to the original M-file, but `encrypt` has scrambled its contents to make it unreadable. Use the optional input argument `-inplace` to make `encrypt` store each MC-file in the same directories as the corresponding M-file.

Chemical Engineering Module Updates

Updated Rising Bubble Model Using Level Sets

A new version of this model includes several improvements for easier modeling and improved accuracy. This version also corrects some errors in the documentation. The following changes are the most important:

- To avoid computing the curvature explicitly in the surface-tension force, you integrate this force by parts (after multiplying with the test functions) using a surface divergence theorem. By doing so you can apply the force by adding the weak contribution

$$\sum_{i=1}^{d} \sigma\delta(\nabla_s \hat{\mathbf{u}}_i)_i$$

  where $\nabla_s = (\mathbf{I} - \mathbf{n}\mathbf{n}^T) \cdot \nabla$ represents the surface gradient operator, and $\hat{\mathbf{u}}_i$ is the test function for the ith velocity component. This way of computing the surface-tension force results in better accuracy than the explicit application of the force because it uses only first-order derivatives (of the unknowns).
- The model uses a Heaviside function for a smooth transition of the density at the interface between the fluids. This function now takes a normalized level set function as its first input argument and the scale is a mesh-dependent expression. This makes the Heaviside function work across a wider range and for different meshes.
- The added diffusivity at the phase interface is now also defined using a mesh-dependent expression.

This model is included on the COMSOL 3.2b CDs and is available for download at http://www.comsol.com/support/updates/. Full model documentation is available in the PDF version of these Release Notes at http://www.comsol.com/support/updates/ and on the COMSOL 3.2b CD 1.

2

COMSOL Multiphysics 3.2b Model Library Update

This chapter contains the following updated models for the COMSOL Multiphysics 3.2v Model Library:

- Shallow Water Equations
- Sloshing Tank

Shallow Water Equations

The shallow water equations are frequently used for modeling both oceanographic and atmospheric fluid flow. Models of such systems lead to the prediction of areas eventually affected by pollution, coastal erosion, and polar ice-cap melting.

Comprehensive modeling of such phenomena using physical descriptions such as the Navier-Stokes equations can often be problematic, due to the scale of the modeling domains as well as through resolving free surfaces. The shallow water equations, of which there are a number of representations, provide an easier description of such phenomena.

This 1D model investigates the settling of a wave over a variable bed as a function of time. The initial wave and the shape of the bed are represented by mathematical relations so that it is easy to change parameters such as the wave amplitude or the bed's shape.

Introduction

This example uses the *Saint-Venant's* shallow water equations, which are the following:

$$\frac{\partial z}{\partial t} + \nabla \cdot (zv) = 0$$

and

$$\frac{\partial (zv)}{\partial t} + \nabla \cdot (zvv^T) + gz\nabla z_s - \nu\Delta(zv) = 0$$

where z is the thickness of the water layer (m), v is the velocity (m s^{-1}), g the gravity constant (m s^{-2}), and ν the kinematic viscosity (m^2 s^{-1}). The definition of the

thickness of the water layer, z, is z_s–z_f, where z_s and z_f are the measures in Figure 2-1 below.

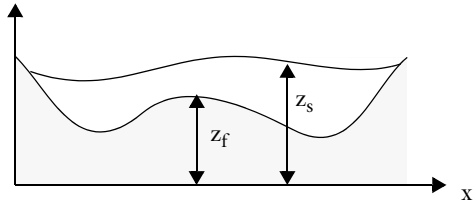


Figure 2-1: Representative vertical section through the fluid domain showing the bed of a lake and the water surface.

Artificial Stabilization

With time, the flow develops discontinuities known as hydraulic jumps. Use artificial stabilization to replace the jumps by steep fronts that can be resolved on the grid. Small amplitude waves on still water of depth z move with velocity $\sqrt{gz}$. The maximal propagation velocity is $v_{phase} = |v| + \sqrt{gz}$ for water waves.

Stabilize the solution by adding artificial viscosity chosen to make the cell Reynolds number of order unity. Add the term $tune \cdot v_{phase} \cdot h \cdot \frac{\partial}{\partial x}(vz)$ to the physical viscous momentum flux $\nu \cdot \frac{\partial}{\partial x}(zv)$. *tune* is a $O(1)$ tuning parameter and h is the local element size. The contribution is added to the *div*-term of the conservation law so that it does not affect the shock speeds. The modification is first order in element size.

Model Definition

This model studies a simple example of shallow water in a channel with bottom topography shown in Figure 2-2. Notice the difference in scale between the *x*- and *y*-directions.

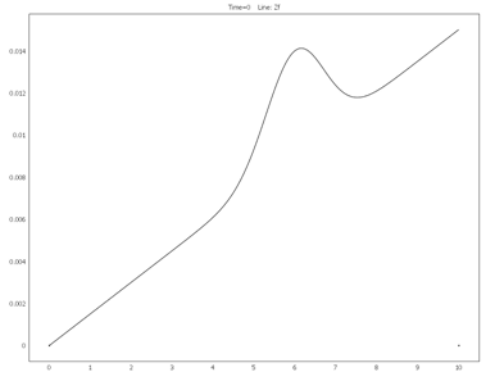


Figure 2-2: Sea bed profile, z_f, used in the model.

Dirichlet boundary conditions ($v = 0$) are implemented at both ends, while the physics are described by the equations above. The initial condition is a wave profile, which the following expression defines:

$$z_0 = 2 \cdot 10^{-2} - z_f + 5 \cdot 10^{-3} e^{\frac{-(x-3)^2}{1^2}}$$

where z_f is the analytical expression for the sea bed profile (see Figure 2-2). The elevation of the water surface is $z + z_f$, while Figure 2-3 shows $z_0 + z_f$.

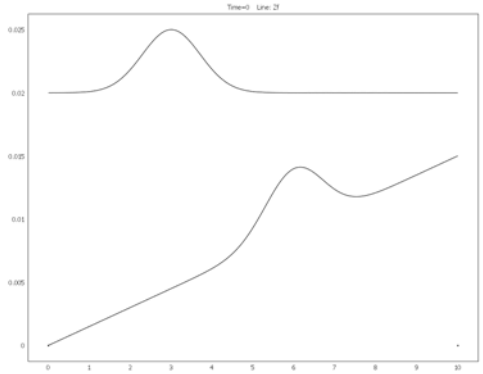


Figure 2-3: The initial water surface profile, $z_0 + z_f$, and the sea bed profile, z_f.

Results and Discussion

The simulation runs for 60 seconds. Figure 2-4 shows the water surface and slope of the sea bed at six output times toward the beginning of the simulation.

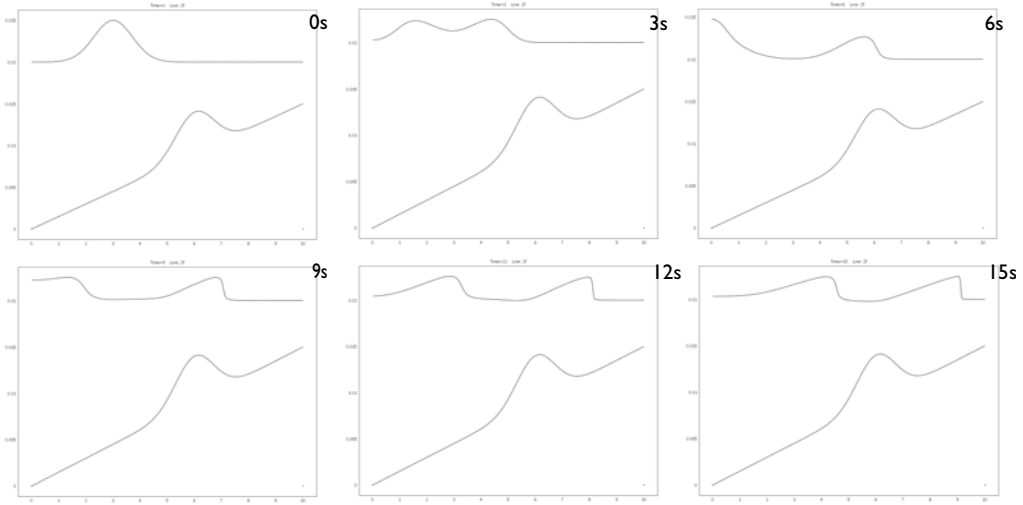


Figure 2-4: The water level and the slope of the sea bed at six output times. Time spans from t = 0 to t = 15 at steps of 3 seconds.

The simulation clearly shows the influence of the topography of the sea bed on the elevation of the water surface. Another interesting visualization of the results is an animation, which is easy to create using COMSOL Multiphysics.

Modeling in COMSOL Multiphysics

The modeling procedure is straightforward using the PDE, General Form application mode with two dependent variables: `Z` and `ZV`. It is easy to define expressions, such as the one that describes the initial wave profile, z_0, in the appropriate dialog box.

Reference

1. O. Pironneau, *Finite Element Methods for Fluids*, John Wiley & Sons, 1989.

Modeling Using the Graphical User Interface

1 Double-click the **COMSOL Multiphysics** icon to open the **Model Navigator**.

2 Select **1D** in the **Space dimension** list.

3 Select **COMSOL Multiphysics>PDE Modes>PDE, General Form>Time-dependent analysis**.

4 Type `Z ZV` in the **Dependent variables** edit field.

5 Click **OK**.

OPTIONS AND SETTINGS

1 Select **Constants** in the **Options** menu.

2 Enter the following constants:

NAME	EXPRESSION
nu1	1e-6
ge	9.8
x0	6
a	0.005
k1	0.0015
tune	0.1

3 Click **OK**.

4 Select **Expressions>Scalar Expressions** in the **Options** menu.

5 Enter the following scalar expressions:

NAME	EXPRESSION
Zf	a*exp(-(x-x0)^2)+k1*x
dZfdx	(-2*x+2*x0)*a*exp(-(x-x0)^2)+k1
Zs	Z+Zf
Z0	0.02-Zf+0.005*exp(-(x-3)^2/1^2)
vphase	abs(ZV/Z)+sqrt(ge*Z)
nu	nu1+vphase*h*tune

6 Click **OK**.

GEOMETRY MODELING

1 Select **Specify Objects>Line** in the **Draw** menu.

2 Type `0 10` in the **x** edit field. This creates a line from 0 to 10 along the *x* axis.

3 Click **OK**.

4 Click the **Zoom Extents** button in the main toolbar.

PHYSICS SETTINGS

Subdomain Settings

1 Select **Subdomain Settings** in the **Physics** menu.

2 Select subdomain 1 in the **Subdomain selection** list.

3 Type `ZV` in the first edit field (first row) in the **Flux vector** area.

4 Type `ZV*ZV/Z-nu*ZVx` in the second edit field in the **Flux vector** area.

5 Click the **F** tab.

6 Type `0` in the first edit field in the **Source** area.

7 Type `-ge*Z*(Zx+dZfdx)` in the second edit field in the **Source** area.

8 Click the **Init** tab to set the initial conditions.

9 Type `Z0` in the **Z(t_0)** edit field. Type `0` in all other edit fields.

10 Click **OK**.

Boundary Settings

1 Select **Boundary Settings** from the **Physics** menu.

2 Select both boundary 1 and 2 in the **Boundary selection** list.

3 Click the **R** edit field.

4 Type `0` in the first edit field (first row).

5 Type `-ZV` in the second edit field.

6 Click **OK**.

MESH GENERATION

1 Select **Mesh parameters** in the **Mesh** menu.

2 Type `0.05` in the **Maximum element size** edit field.

3 Click **Remesh** and **OK**.

COMPUTING THE SOLUTION

1 Select **Solver Parameters** in the **Solve** menu.

2 Type `linspace(0,60,61)` in the **Times** edit field in the **Time Stepping** area.

3 Type `1e-5` in the **Relative Tolerance** edit field in the **General** area.

4 Type `1e-7` in the **Absolute Tolerance** edit field in the **General** area.

5 Click **OK**.

6 Click the **Solve** button.

POSTPROCESSING AND VISUALIZATION

The most interesting part of the results is obtained by looking at the scalar expression Zs, corresponding to the surface topography, at different times compared to the topography of the bottom.

1 Click the **Postprocessing** button.

2 Click the **Line** tab.

3 Type `Zs` in the **Expression** edit field.

4 Click **Apply**.

5 Click the **General** page.

6 Select `6` in the **Solution at time** drop-down list.

7 Check the **Keep current** plot.

8 Click the **Line** tab.

9 Type `Zf` in the **Expression** edit field.

10 Click **OK**.

Sloshing Tank

Introduction

This 2D model demonstrates the ability of COMSOL Multiphysics to simulate dynamic free surface flow with the help of a moving mesh. The study models fluid motion with the incompressible Navier-Stokes equations. The fluid is initially at rest in a rectangular tank. The motion is driven by the gravity vector swinging back and forth, pointing up to 4 degrees away from the downward y direction at its extremes.

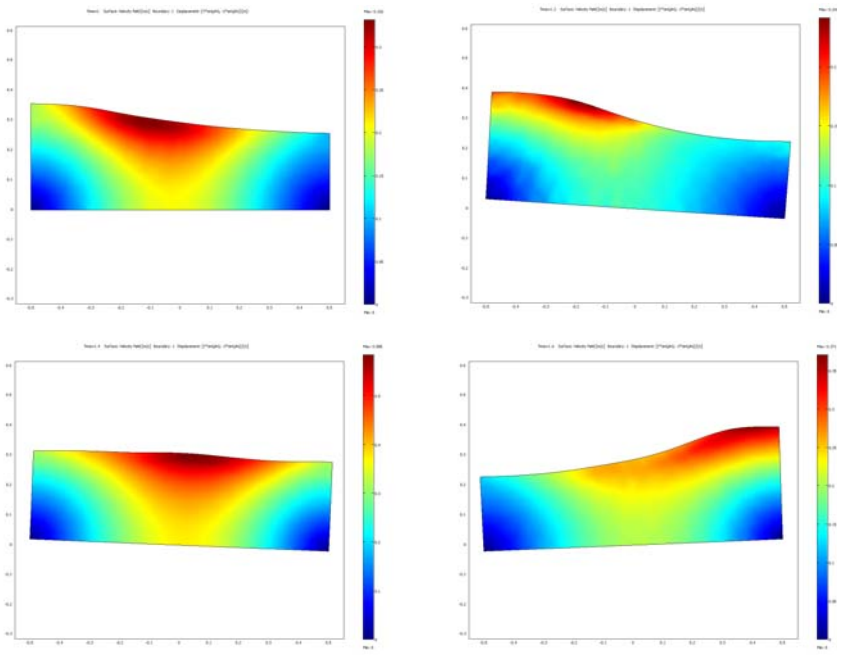


Figure 2-5: Snapshots of the velocity field at t = 1 s, t = 1.2 s, t = 1.4 s, and t = 1.6 s. The inclination of the gravity vector is indicated by the leaning of the tank.

Because the surface of the fluid is free to move, this model is a nonstandard computational task. The ALE (arbitrary Lagrangian-Eulerian) technique is, however, well suited for addressing such problems. Not only is it easy to set up using the Moving Mesh (ALE) application mode in COMSOL Multiphysics, but it also has the advantage that it represents the free surface boundary with a domain boundary on the moving mesh. This allows for the accurate evaluation of surface properties such as

curvature, making surface tension analysis possible. Note, however, that this example model neglects surface tension effects.

Model Definition

DOMAIN EQUATIONS

This model describes the fluid dynamics with the incompressible Navier-Stokes equations:

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho(\mathbf{u}\cdot\nabla)\mathbf{u} - \nabla\cdot(-p\mathbf{I} + \eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)) = \mathbf{F}$$

$$\nabla\cdot\mathbf{u} = 0$$

where ρ is the density, $\mathbf{u} = (u, v)$ is the fluid velocity, p is the pressure, $\mathbf{I}$ is the unit diagonal matrix, η is the viscosity, and $\mathbf{F}$ is the volume force. In this example model, the material properties are for glycerol: η = 1.49 Pa s, and ρ = 1.27e3 kg/m^3. The gravity vector enters the force term as

$$F_x = \rho g \sin(\phi_{max}\sin(2\pi f t))$$
$$F_y = -\rho g \cos(\phi_{max}\sin(2\pi f t))$$

where g = 9.81 m/s^2, ϕ_{max} = 4*π/180, and f = 1 s^{-1}.

With the help of the Moving Mesh (ALE) application mode, you can solve these equations on a freely moving deformed mesh, which constitutes the fluid domain. The deformation of this mesh relative to the initial shape of the domain is computed using Winslow smoothing. For more information, please refer to "The Moving Mesh Applcation Mode" on page 303 in the *COMSOL Multiphysics Modeling Guide*. COMSOL Multiphysics takes care of the transformation of the Navier-Stokes equations to the formulation on the moving mesh.

BOUNDARY CONDITIONS FOR THE FLUID

There are two types of boundaries in the model domain. Three solid walls, that are modeled with slip conditions, and one free boundary (the top boundary). The slip boundary condition for the Navier-Stokes equations is

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

where $\mathbf{n} = (n_x, n_y)^T$ is the boundary normal. To enforce this boundary condition, select the Slip/Symmetry boundary condition in the Incompressible Navier-Stokes

application mode. Because the normal vector depends on the degrees of freedom for the moving mesh, a constraint force would act not only on the fluid equations but also on the moving mesh equations. This effect would not be correct, and one remedy is to use non-ideal weak constraints. Ideal weak constraints (the other type of weak constraints) do not remove this effect of the constraint force. For more information about weak constraints, see "Using Weak Constraints" on page 491 in the *COMSOL Multiphysics User's Guide*. The Incompressible Navier-Stokes application mode does not make use of weak constraints per default, so you need to activate the non-ideal weak constraints.

The following weak expression, which you add to the model, enforces the slip boundary condition without a constraint force acting on the moving mesh equations:

$$\hat{\lambda}(\mathbf{u} \cdot \mathbf{n}) - \lambda(\hat{\mathbf{u}} \cdot \mathbf{n}) \tag{2-1}$$

for some Lagrange multiplier variable λ. Here $\hat{\lambda}$ and $\hat{\mathbf{u}}$ denote test functions. See the step-by-step instructions later in this model documentation for details.

The fluid is free to move on the top boundary. The viscous stress in the surrounding environment is neglected. Therefore the stress continuity condition on the free boundary reads

$$(-p\mathbf{I} + \eta(\nabla\mathbf{u} + (\nabla\mathbf{u})^T)) \cdot \mathbf{n} = -p_0\mathbf{n}$$

where p_0 is the surrounding (constant) pressure and η the viscosity in the fluid. Without loss of generality, $p_0 = 0$ for this model. In the Incompressible Navier-Stokes application mode you enforce this boundary condition by selecting Neutral.

BOUNDARY CONDITIONS FOR THE MESH

In order to follow the motion of the fluid with the moving mesh, it is necessary to (at least) constrain the mesh motion to the fluid motion normal to the surface. It turns out that for this type of free surface motion, it is important to not constrain the mesh motion to the fluid motion in the tangential direction. If you would do so, the mesh soon becomes so deformed that the solution no longer converges. The boundary condition for the mesh equations on the free surface is therefore

$$(x_t, y_t)^T \cdot \mathbf{n} = \mathbf{u} \cdot \mathbf{n}$$

where $\mathbf{n}$ is the boundary normal and $(x_t, y_t)^T$ the velocity of mesh (see "Mathematical Description of the Mesh Movement" on page 290 in the *COMSOL Multiphysics Modeling Guide*). In the Moving Mesh (ALE) application mode, you specify this

boundary condition by selecting the tangent and normal coordinate system in the deformed mesh and by specifying a mesh velocity in the normal direction, where you enter the right-hand side expression from above as `u*nx+v*ny`. The Moving Mesh (ALE) application mode uses non-ideal weak constraints per default and for this boundary condition it adds the weak expression

$$\hat{\lambda}(((x_t,y_t)^T - \mathbf{u}) \cdot \mathbf{n}) - \lambda((\hat{x},\hat{y})^T \cdot \mathbf{n})$$

to ensure that there are no constraint forces acting on the fluid equations. Here again, λ denotes some Lagrange multiplier variable (not the same as before) and $\hat{\lambda}$, $\hat{x}$, and $\hat{y}$ denote test functions. There is no need to modify this expression. Choose **Physics>Equation System>Boundary Settings** and select the free boundary (boundary 3) to see how to enter this expression in COMSOL Multiphysics. The expression implies that there is a flux (or force) on the free boundary for the moving mesh coordinate equations $\nabla x \cdot \mathbf{n} = \lambda n_x$ and $\nabla y \cdot \mathbf{n} = \lambda n_y$, respectively. Furthermore, to be able to follow the fluid motion with the mesh motion, the moving mesh must not be constrained in the tangential direction on the side walls. In the Moving Mesh (ALE) application mode, you specify this boundary condition by using the global coordinate system and setting the mesh displacement to zero in the x-direction. At the bottom of the tank the mesh is fixed, which you obtain in a similar way by setting the mesh displacements to zero in both the x- and y-directions.

Results

Figure 2-6 on page 55 shows the tank at a few different points in time. The colors represent the velocity field. Whereas the modeling is set up using a fixed tank and a swinging gravity vector, postprocessing using a deformation plot gives the tank a corresponding inclination. The inclination angle of the tank is exactly the same as the angle of the gravity vector from its initial vertical position.

To illustrate the dynamics in the tank, you can plot the wave height versus time at one of the vertical walls, as in the following plot.

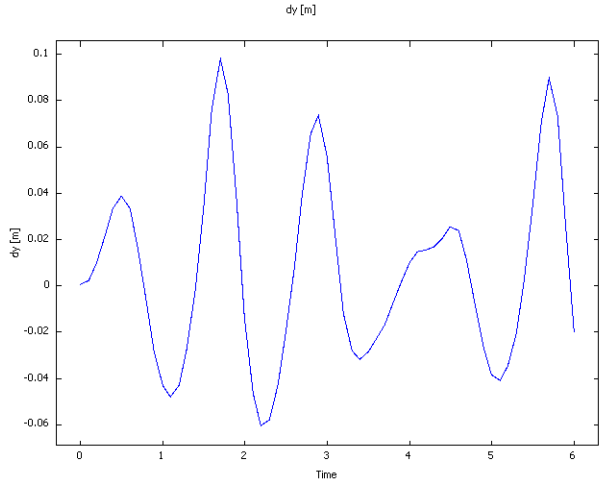


Figure 2-6: Wave height at X = 6.6 m for 0 ≤ t ≤ 20 s.

The movie file that accompanies this model shows the waves in the swinging tank, with a color scale indicating the vorticity.

Model Library path: `COMSOL_Multiphysics/Fluid_Dynamics/sloshing_tank`

Modeling Using the Graphical User Interface

1. Start COMSOL Multiphysics.
2. In the **Model Navigator**, click the **Multiphysics** button.
3. Select **COMSOL Multiphysics>Deformed Mesh>Moving Mesh (ALE)>Transient analysis** and click **Add**.
4. Click the **Application Mode Properties** button and set **Smoothing method** to **Winslow**.
5. Select **COMSOL Multiphysics>Fluid Dynamics>Incompressible Navier-Stokes>Transient analysis** and click **Add**.
6. Click **OK**.

GEOMETRY MODELING

1 Shift-click the **Rectangle/Square** button in the Draw toolbar.

2 Specify the rectangle settings according to the table below

PROPERTY	EXPRESSION
Width	1
Height	0.3
Position: Base	Corner
Position: x	-0.5
Position: y	0

3 Click the **Zoom Extents** button in the Main toolbar.

OPTIONS AND SETTINGS

1 Open the **Constants** dialog box from the **Options** menu and enter the following constants. The descriptions are optional.

NAME	EXPRESSION	DESCRIPTION
rho	1270	Glycerol density (kg/m^3)
nu	1.49	Glycerol viscosity (Pa*s)
phi_max	4*pi/180	Maximum angle of inclination (rad)
freq	1	Frequency (Hz)
g	9.81	Acceleration due to gravity (m/s^2)

2 From the **Options** menu, choose **Expressions>Scalar Expressions**.

3 Enter the following scalar expressions:

NAME	EXPRESSION	DESCRIPTION
phi	phi_max*sin(2*pi*freq*t)	Angle of inclination (rad)
grav_x	g*sin(phi)	Gravity vector x component (m/s^2)
grav_y	-g*cos(phi)	Gravity vector y component (m/s^2)

PHYSICS SETTINGS

Properties

1 In the Incompressible Navier-Stokes application mode, choose **Properties** from the **Physics** menu.

2 In the **Application Mode Properties** dialog box, select **Non-ideal** in the **Weak constraints** list and then click **OK**.

Subdomain Settings

Open the **Subdomain Settings** dialog box and apply the settings in the table below.

SUBDOMAIN	1
ρ	rho
η	nu
F_x	grav_x*rho
F_y	grav_y*rho

Boundary Conditions

1 Open the **Boundary Settings** dialog box from the **Physics** menu and enter boundary conditions according to the table below.

BOUNDARY	1, 2, 4	3
Boundary Condition	Slip/Symmetry	Neutral

2 Click **OK**.

3 Now, the weak non-ideal constraint implementing the slip/symmetry condition is not correct. Open the **Boundary Settings** dialog box from the **Physics>Equation Settings** menu and on boundaries 1, 2, and 4, change the expression for the Lagrange multiplier variable `lm3` in the eighth row on the **weak** tab to the following:

```
lm3_test*(u*nx_ns+v*ny_ns)-lm3*(u_test*nx_ns+v_test*ny_ns)
```

This is Equation 2-1 for the slip boundary condition without a constraint force. There are as many rows for weak terms as there are variables on the boundary (including Lagrange multipliers). You can use any row to enter a weak term, but it is easiest to modify the row where the default term appears.

4 Click **OK**.

5 Go to the **Multiphysics** menu and select **Moving Mesh (ALE)**.

6 In the **Boundary Settings** dialog box, apply the following boundary conditions for the mesh displacements (only tangential movements on the sides and a fixed mesh at the bottom):

BOUNDARY	1, 4	2
dx	0	0
dy		0

7 On boundary 3, select **Tangent and normal coord. sys. in deformed mesh** in the **Coord .sys.** list. Then click the **Mesh velocity** button and type `u*nx+v*ny` in the **vn** edit field to specify the normal mesh velocity as $\mathbf{u} \cdot \mathbf{n}$.

8 On the **Weak Constr.** tab of the **Boundary Settings** dialog box, clear the **Use weak constraints** check box on boundaries 1, 2, and 4. The strong constraints that you specified in the previous step are sufficient on these boundaries. Leave it selected on boundary 3.

9 Click **OK** to close the dialog box.

MESH GENERATION

Click the **Mesh** button to initialize the mesh.

COMPUTING THE SOLUTION

1 Open the **Solver Parameters** window from the **Solve** menu.

2 Select **Time dependent** from the **Solver** list. Enter `0:0.1:6` in the **Times** edit field.

3 On the **Time Stepping** tab, select **Exclude algebraic** in the **Error estimation strategy** list. This excludes the pressure and the moving mesh variables from the error estimation. The equations for those variables do not include time derivatives and become algebraic when solving the equation system using the method of lines.

4 Click **OK**.

5 Click the **Solve** button in the Main toolbar.

POSTPROCESSING AND VISUALIZATION

The default plot shows the *x* component of the moving mesh deformation, in the spatial frame.

1 To plot the velocity field of the glycerol instead, go to the **Surface** tab in the **Plot Parameters** dialog box and select **Velocity field (ns)** from the list of expressions.

2 On the **General** page, clear the **Geometry edges** check box. Click **Apply** to see the plot and use the **Solution at time** list on the **General** tab to browse through the output times.

It is possible to visualize the inclination of the tank by clever use of the deformation plot feature.

3 On the **Deform** page, select the **Deformed shape plot** check box and Set the **Scale factor** to `1`. Enter `Y*sin(phi)` for the X component and `-X*sin(phi)` for the Y component on the **Subdomain Data** tab.

4 Still on the **Deform** page, click the **Boundary Data** tab. Once again, enter `Y*sin(phi)` for the X component and `-X*sin(phi)` for the Y component.

5 On the **Boundary** tab, select the **Boundary plot** check box. Enter 1 in the **Expression** edit field. Select to use a **Uniform** color and pick a black color using the **Color** button.

6 To get a more liquid-looking plot, you may want to go to the **Surface** page and set the **Colormap** to **bone**.

7 Click **Apply** to see the plot.

8 To see the waves in action, go to the **Animate** tab and click **Start Animation**.

To get a more comprehensive overview of the sloshing, you can plot the *y* displacement from equilibrium in a point.

9 Open the **Domain Plot Parameters** dialog box from the **Postprocessing** menu.

10 On the **Point** tab, select point 4 from the **Point selection** list.

11 Enter `dy_ale` in the **Expression** edit field and click **OK** to see the plot.

3

Electromagnetics Module 3.2b Model Library Update

This chapter contains an updated model for the Electromagnetics Module 3.2b Model Library:

- Inductance of a Power Inductor

Inductance of a Power Inductor

The model shows an inductance calculation on a large 3D geometry using higher-order vector elements and memory-efficient iterative solver settings.

Introduction

Power inductors are a central part of many low-frequency power applications. They are, for example, used in switched power supplies and DC-DC converters. The inductor is used in conjunction with a high power semiconductor switch that operates at a certain frequency, stepping up or down the voltage on the output. The relative low voltage and high power consumption puts high demands on the design of the power supply and especially on the inductor, which must be designed with respect to switching frequency, current rating, and warm environments.

A power inductor usually has a magnetic core to increase its inductance value, reducing the demands for a high frequency while keeping the sizes small. The magnetic core also reduces the electromagnetic interference with other devices. There are only crude analytical formulas or empirical formulas available for calculating impedances, so computer simulations or measurements are necessary in the design of these inductors. This model uses a design from an external CAD software, imports the geometry to COMSOL Multiphysics, and finally calculates the inductance from the specified material parameters and frequency.

Model Definition

The model uses the Quasi-static application mode taking electric induced and inductively induced currents into account. This formulation, often referred to as an A/V formulation, solves both for the magnetic vector potential $\mathbf{A}$ and the electric potential V. In addition, it must also solve for the Gauge fixing on the vector potential,

$$\nabla \cdot \mathbf{A} = 0$$

At low frequencies the inductance is almost constant. For high frequencies the capacitive effects play a role, and the permeability usually decreases, causing a frequency-dependent inductance. A model limiting factor in increasing the frequency is the skin depth, so at a frequency in the vicinity of 10 kHz the model either needs a

finer mesh at the conductor boundary or an impedance boundary condition. The following table lists the material properties used in this model.

MATERIAL PARAMETER	COPPER	CORE
σ	5.997e7	10
ε_r	1	1
μ_r	1	1e3

Using a low conductivity for the surrounding air improves the stability of the iterative solver. This has negligible impact on the solution.

The outer boundaries are mainly magnetic insulation and electric insulation,

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

For the boundaries to the conductor, one end is grounded, and the other end has a port boundary condition. The port boundary condition gives the impedance of the inductor, and you can calculate the inductance from the formula,

$$L_{11} = \frac{Im(Z_{11})}{\omega}$$

where ω is the angular frequency, and $Im(Z_{11})$ is the imaginary part of the impedance.

COMPUTING THE SOLUTION

An analysis using the AV formulation with Gauge fixing turned on consumes large amounts of memory when you use direct solvers like UMFPACK or SPOOLES. This model has almost 200 000 degrees of freedom, so an iterative solver is necessary. The geometric multigrid (GMG) solver with the Vanka pre- and postsmoother solves this problem efficiently. Different element order defines the multigrid hierarchy, which means that a direct solver solves the problem on linear vector and Lagrange elements. The iterative solver then produce the solution for the quadratic versions of these elements.

Results and Discussion

At a frequency of 1 kHz the inductance is 97 μH, and the figure below shows the electric potential and the magnetic flux density in a combined plot.

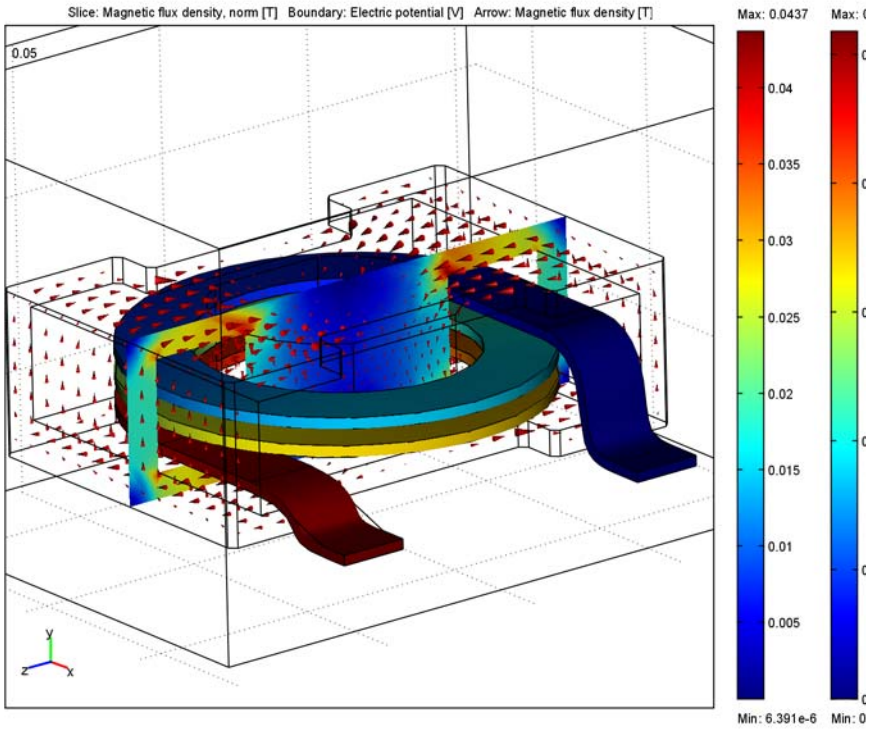


Figure 3-1: The final plot of the power inductor, showing the potential on the coil, the magnitude of the flux density inside the ferrite core, and the direction of the same as arrows.

Modeling Using the Graphical User Interface

MODEL NAVIGATOR

1 Select **3D** in the **Space dimension** list.

2 In the list of application modes, select **Electromagnetics Module>Quasi-Statics, Electromagnetic>Electric and Induction Currents**.

3 From the **Element** list, select **Vector, Lagrange - Quadratic**.

4 Click **OK**.

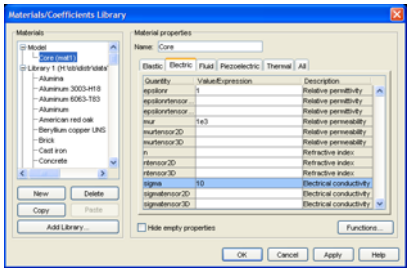
OPTIONS AND SETTINGS

1 From the **Options** menu, choose **Material/Coefficient Library**. In the dialog box that appears, click the **New** button.

2 Type `Core` in the **Name** edit field for the new material.

3 Click the **Electric** tab and type the values for each material parameter listed in the table below.

MATERIAL PARAMETER	CORE
σ	10
ε_r	1
μ_r	1e3



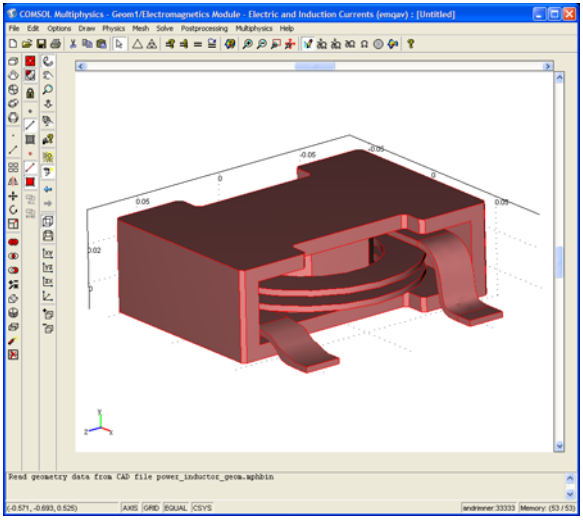
4 Click **OK**.

GEOMETRY MODELING

1 From the **File** menu, choose **Import>CAD Data From File**.

2 In the **Import CAD Data From File** dialog box, make sure that the **COMSOL Multiphysics file** or **All 3D CAD files** is selected in the **Files of type** list.

3 Locate the `power_inductor_geom.mphbin` file and click **Import**.



4 From the **Draw** menu, choose **Block**.

5 In the dialog box that appears, define the block properties according to the table below.

LENGTH X	LENGTH Y	LENGTH Z	BASE	AXIS BASE POINT (X,Y,Z)
0.15	0.12	0.2	Corner	(-0.07,-0.031,-0.1)

PHYSICS SETTINGS

Scalar Variables

1 Open the **Scalar Variables** dialog box from the **Physics** menu.

2 Type 1e3 for the frequency, **nu_emqav**. Click **OK**.

Boundary Conditions

1 Open the **Boundary Settings** dialog box from the **Physics** menu.

2 Select all boundaries and make sure that the default **Magnetic insulation** is selected from the **Boundary condition** list.

3 Click the **Electric Parameters** tab. Select **Electric insulation** from the **Boundary condition** list.

4 Select boundary number **78**, which is one of the boundaries for the inductor coil. Choose the **Ground** boundary condition from the **Boundary condition** list.

5 Select the other coil boundary, which is number **79**. Select the **Port** boundary condition.

6 Next, click the **Port** tab, and select the **Use port as input** check box. Leave the other settings at their defaults, which is 1 for **Port number** and **Fixed current density** for **Input property**.

7 Click **OK**.

Subdomain Settings

1 Open the **Subdomain Settings** dialog box from the **Physics** menu.

2 Select subdomain **1** and click **Electric Parameters** tab. Type 1 in the edit field for the electric conductivity.

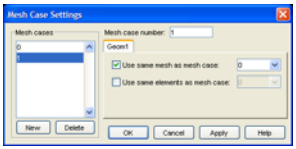
3 Select subdomain **2** and select **Core** from the **Materials** list. This is the material you defined in the Materials library earlier.

4 Select subdomain **3** and click the **Load** button. Locate and select **Copper** in the dialog box and click **OK**.

5 Click **OK** again to close the **Subdomain Settings** dialog box.

MESH GENERATION

1. Open the **Mesh Parameters** dialog box from the **Mesh** menu.
2. Select the **Coarse** mesh size from the **Predefined mesh sizes** list.
3. Type `1.8` in the **Mesh element growth rate** edit field, `0.4` in the **Mesh curvature factor** edit field, and type `0.02` in the **Mesh curvature cut off** edit field.
4. Click **Remesh** and then click **OK**.
5. From the **Mesh** menu, choose **Mesh Cases**.
6. In the **Mesh Case Settings** dialog box, click the **New** button and select the **Use same mesh as mesh case** check box. Make sure that the list to the right has mesh case number **0** selected.



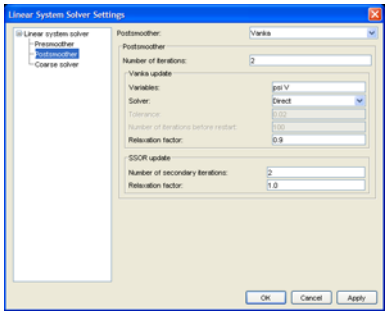
7. Click **OK**.
8. Open the **Subdomain Settings** dialog box from the **Physics** menu.
9. Click the **Element** tab, select all subdomains, and select **Vector, Lagrange - Linear** from the **Predefined elements** list. Linear elements are now used for mesh case 1, and quadratic elements for mesh case 0, both defined on the same mesh.
10. Click **OK**.

COMPUTING THE SOLUTION

The program can use the linear order element combination for a coarse solution and solve for the quadratic elements using the geometric multigrid solver. The linear solution is then used in the preconditioning step.

1. Open the **Solver Parameters** dialog box from the **Solve** menu.
2. Select **Geometric multigrid** from the **Linear system solver** list.
3. Click the **Settings** button. In the dialog box that appears, make sure that Linear system solver is selected in the field to the left. Select **Manual** from the **Hierarchy generation method** list.

4 Click the **Presmoother** item in the tree view. Type `psi V` in the **Variables** edit field. Do the same for the **Postsmoother** item.



5 All other settings can be left at their default. The default settings are described in more detail in "Solving Large 3D Wave Problems" in Chapter 2 of the *Electromagnetics Module User's Guide*.

6 Click **OK** to close the **Linear System Solver Settings** dialog, and then click **OK** in the **Solver Parameters** dialog box.

7 Click the **Solve** button.

POSTPROCESSING AND VISUALIZATION

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

2 Make sure that the **Slice**, **Boundary**, **Arrow**, and **Geometry edges** check boxes are selected under the **General** tab.

3 Click on the **Slice** tab, and select **Magnetic flux density, norm** from the **Predefined quantities** list.

4 Type `1` in the **x levels** edit field.

5 Click on the **Boundary** tab, and select **Electric potential** from the **Predefined quantities** list.

6 Click the **Arrow** tab, and select **Magnetic flux density** from the **Predefined quantities** list.

7 In the **x points**, **y points**, and **z points** edit fields for the **Number of points**, type `20`, `7`, and `20`, respectively.

8 Select the **Cone** in the **Arrow type** list.

9 Click **OK**.

10 It is now necessary to remove some boundaries and subdomains from the plot. From the **Options** menu choose **Suppress>Boundaries**. Select all boundaries that are

part of the coil. Click **Apply**, then click the **Invert Suppression** button, and finally click **Cancel**.

11 From the **Options** menu, choose **Suppress>Subdomains**. Select subdomain 1 and click **OK**.

12 Click on the **Postprocessing mode** button, and the plot in Figure 3-1 on page 64 should appear after proper rotation and zoom operations.

13 To get the inductance value, choose **Data Display>Global** from the **Postprocessing** menu. Type `imag(Z11_emqav)/omega_emqav` in the **Expression to evaluate** edit field.

You should get a value close to 97 μH in the message field when you click **OK**.

4

Structural Mechanics Module 3.2b Model Library Update

This chapter contains the following new and updated models for the Structural Mechanics Module 3.2b Model Library:

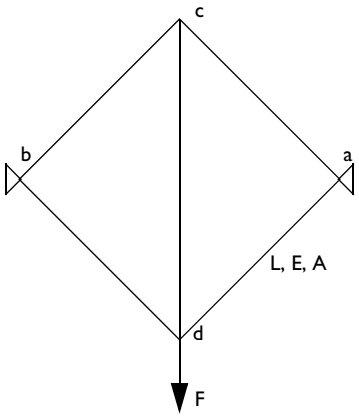
- In-Plane Truss—Example model for the new Truss application mode
- Coupled Vibration—Updated model
- Traffic Tunnel—Updated model

Building and Solving an In-Plane Truss Model

In the following example you build and solve a simple 2D truss model using the In Plane Truss application mode. This model calculates the deformation of a simple geometry. The example is based on problem 11.1 in *Aircraft Structures for Engineering Students* by T.H.G Megson (Ref. 1). The results show perfect agreement with the analytical results given in Ref. 1.

Model Definition

The geometry consists of a square symmetrical truss built up by five members. All trusses has the same cross-section area A. The side length is L and the Young's modulus is E.



GEOMETRY

- Truss side length, L=2 m,
- The truss members has a circular cross section with a radius of 0.05 m

MATERIAL

Aluminum Young's modulus, E=70 GPa.

CONSTRAINTS

Displacements in both directions are constrained at *a* and *b*.

LOAD

A vertical force, F of 50 kN is applied at the bottom corner.

Results and Discussion

The following table shows a comparison between the results calculated with COMSOL Multiphysics and the analytical results from Ref. 1.

RESULT	COMSOL MULTIPHYSICS	REF. 1
Displacement at d	-5.15E-4 m	-5.15E-4 m
Displacement at c	-2.13E-4 m	-2.13E-4 m
Axial force in member ac=bc	-10.4 kN	-10.4 kN
Axial force in member ad=bd	25.0 kN	25.0 kN
Axial force in member cd	14.6 kN	14.6 kN

The results are in total agreement.

COMSOL plot visualizing the deformed geometry together with the axial forces in the trusses.

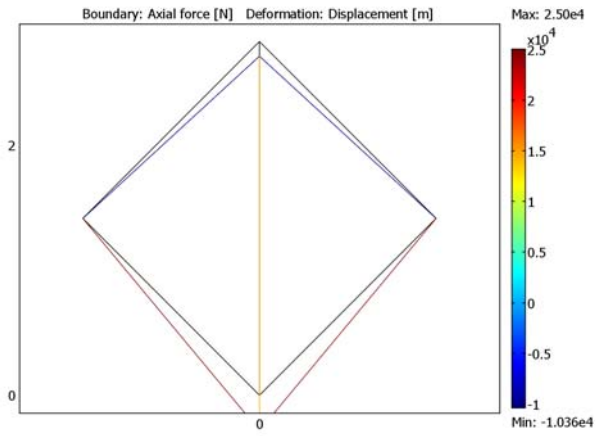
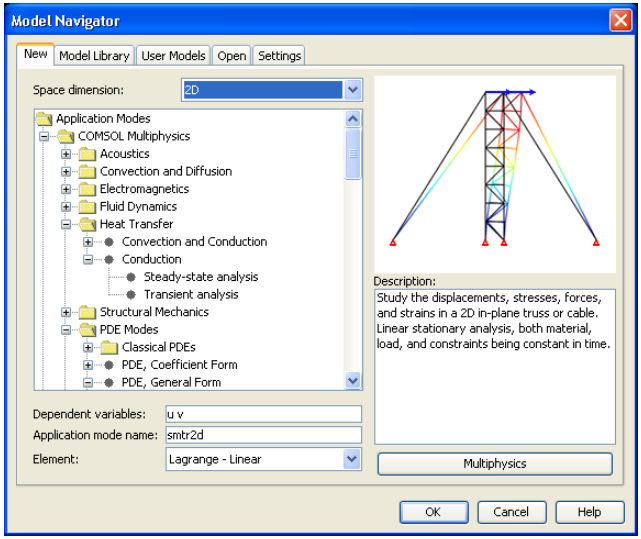


Figure 4-1: Deformed geometry and axial forces.

MODEL NAVIGATOR

1 Select **2D** in the **Space dimensions** list on the **New** page in the **Model Navigator.**

2 Select **Structural Mechanics Module>In-Plane Truss** and click **OK**.



OPTIONS AND SETTINGS

1 From the **Options** menu, choose **Axes/Grid Settings**.

2 Set axis and grid settings according to the following table.

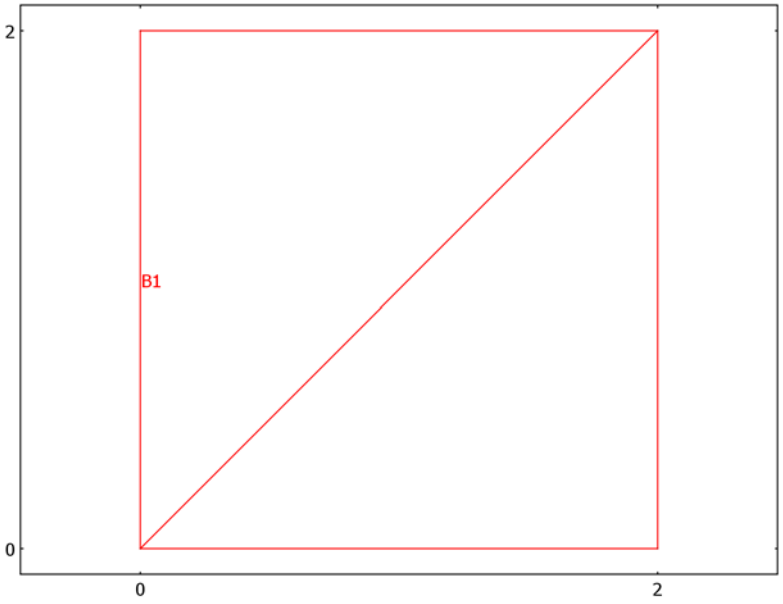
AXIS		GRID	
x min	-4	x spacing	2
x max	4	Extra x	-
y min	-3	y spacing	2
y max	3	Extra y	-

GEOMETRY MODELING

1 Select **Line** from the Draw toolbar.

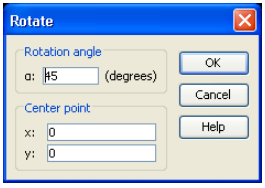
2 Double click on the **SOLID** button in the **Status** bar, to turn off the solid feature when drawing the lines.

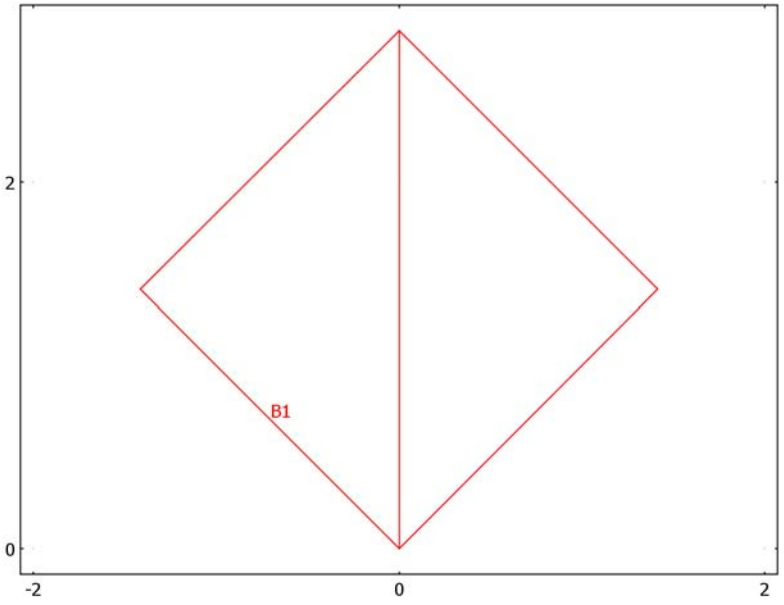
3 Draw a line from (0, 0) through (0, 2), (2, 2), (2, 0), (0, 0), and (2, 2) by clicking on the left mouse button at these coordinates. End the line by clicking on the right mouse button.



4 Click on the **Rotate** button in the **Draw** toolbar to open the **Rotate** dialog box.

5 Enter 45 as the rotating angle and click **OK** to rotate the truss.





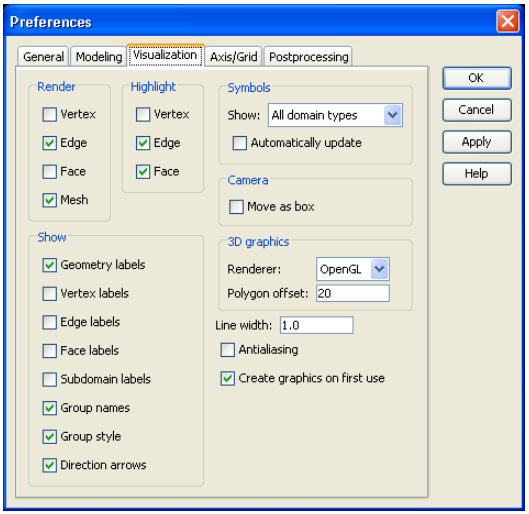
PHYSICS SETTINGS

One very helpful feature when specifying loads and constraints are symbols. Use this feature to see where you have specified constraints and applied forces.

1 Select **Preferences** from the **Options** menu to open the **Preferences** dialog box.

2 Click on the **Visualization** page and select **All domain types** in the **Show** list in the **Symbols** frame.

3 Select **Automatically update** and click **OK** to close the dialog box.



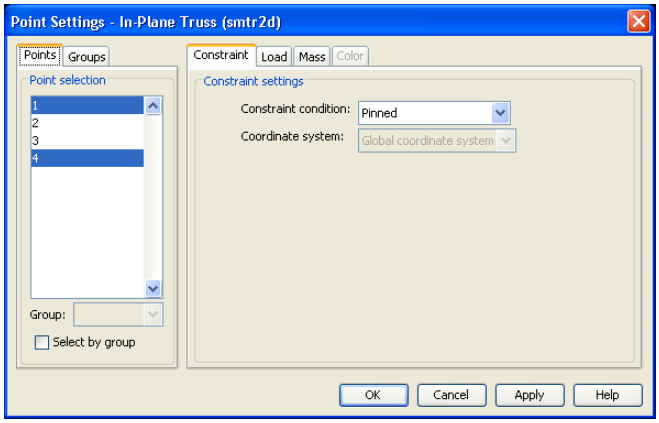
Point Settings

Constrain the x- and y-displacements at the left and right corners of the truss.

1 Select **Point Settings** from the **Physics** menu.

2 Specify constraints on the **Constraint** page according to the following table:

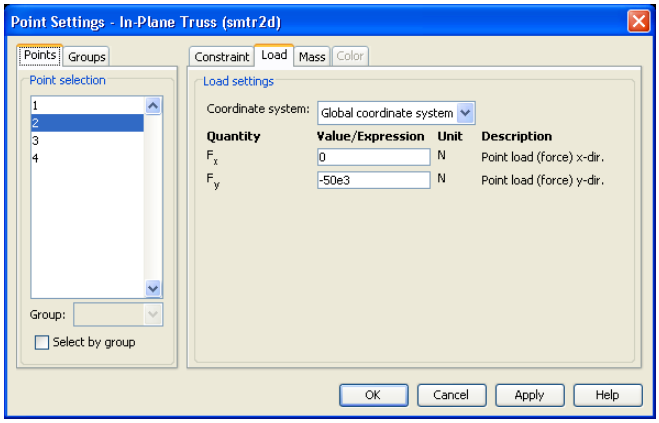
POINT	**1 4**	
Page	Constraint	
	$\mathbf{R}_x$	√
	$\mathbf{R}_y$	√



Specify the vertical force at the bottom corner.

1 Select the **Load** page in the **Point Settings** dialog box.

2 Select point 2 and enter `-50e3` in the $\mathbf{F}_y$ edit field.



3 Click **OK** to close the **Point Settings** dialog box.

Boundary Settings

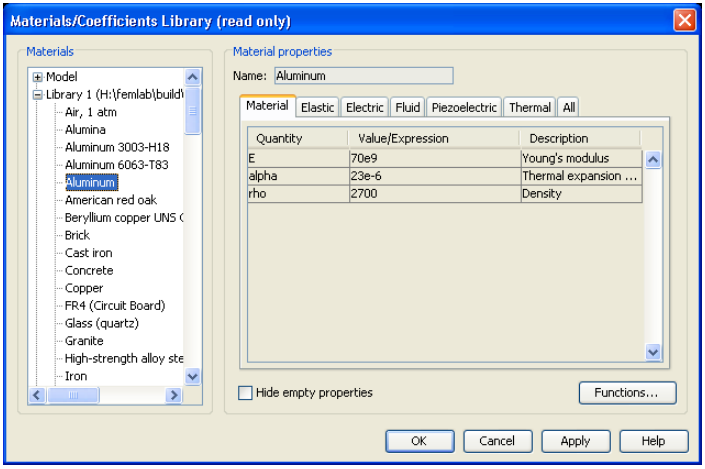
Specify the material and cross section properties of the truss members

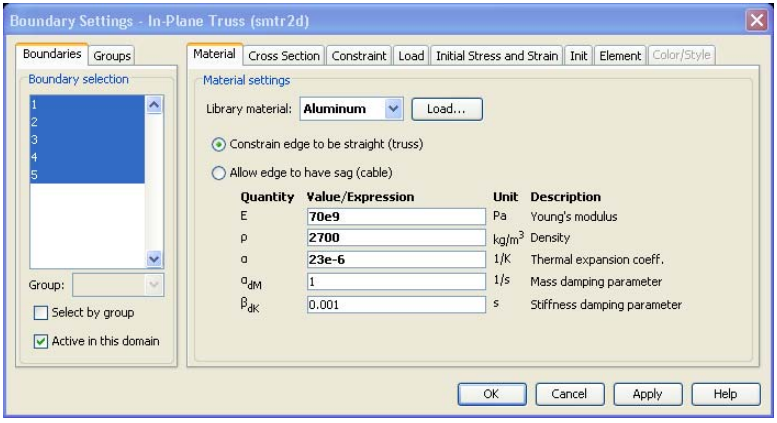
1 Select **Boundary Settings** from the **Physics** menu.

2 Select all five boundaries.

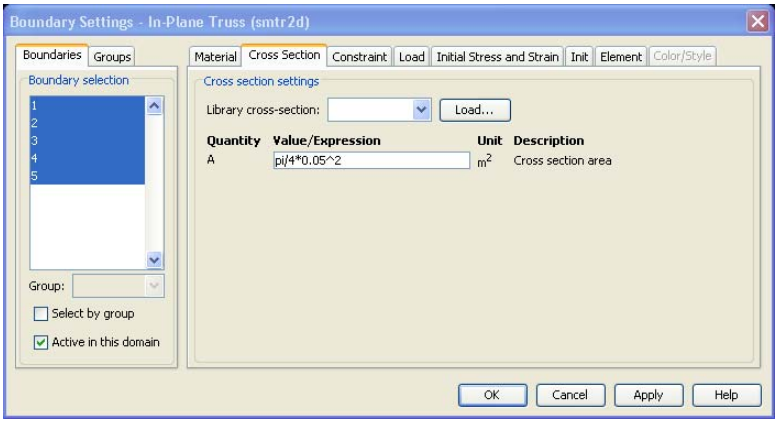
3 Click on the **Load** button on the **Material** page to open the **Material/Coefficients Library** dialog box.

4 Select **Aluminium** from **Library 1** in the **Materials** list and click **OK**.

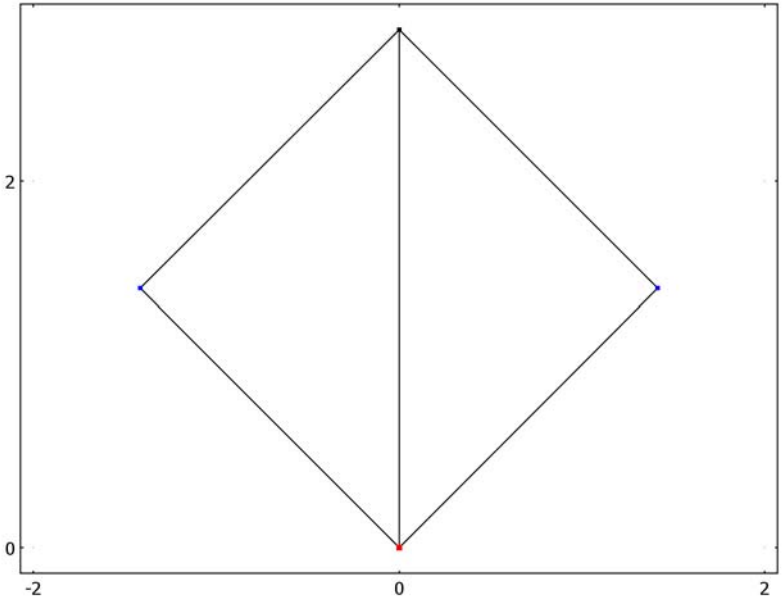




5 Enter `pi/4*0.05^2` in the **Cross section area** (**A**) edit field on the **Cross Section** page.



6 Click **OK** to close the **Boundary Settings** dialog box.



MESH GENERATION

When using the default option **Constrain edge to be straight (truss)** (selected on the **Material** page in the **Boundary Settings** dialog box) the mesh is not critical. The **Allow edge to have sag (cable)** option makes the mesh very critical. The reason for this is that

the internal nodes along the boundary becomes singular as they do not have any stiffness perpendicular to the boundary. To avoid this, use a very coarse mesh with no internal nodes along the boundaries.

Use the default mesh settings as the default option **Constrain edge to be straight (truss)** (selected on the **Material** page in the **Boundary Settings** dialog box) is used in this model.

- Click the **Initialize Mesh** toolbar button to mesh the geometry.

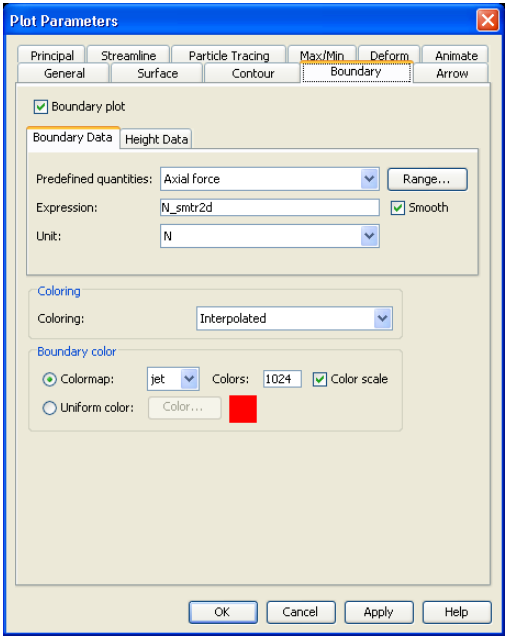
COMPUTING THE SOLUTION

Click the **Solve** toolbar button (**=**) to solve the problem.

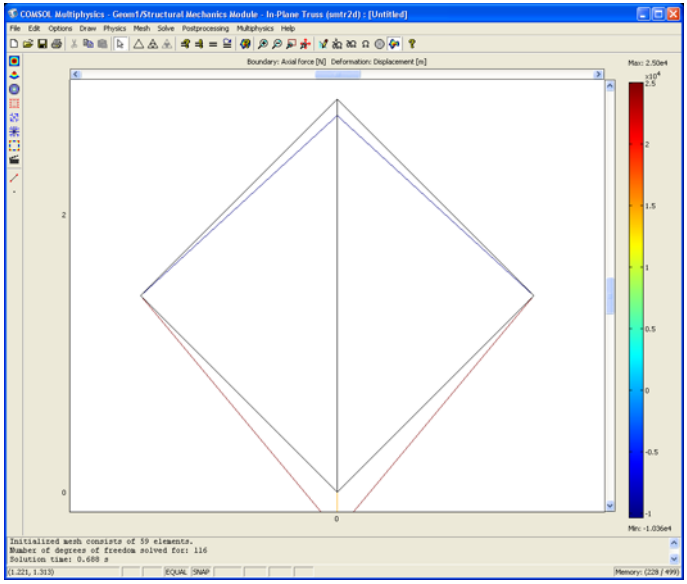
POSTPROCESSING AND VISUALIZATION

Plot the deformed geometry together with the axial forces in the truss members.

1. Select **Plot Parameters** from the **Postprocessing** menu.
2. Check **Deformed shape** in the **Plot type** selection frame on the **General** page.
3. Click on the **Boundary** page.
4. Select **Axial force** from the **Predefined quantities** list.



5 Click **OK** to close the dialog box and view the plot.



Reference

1. T.H.G Megson, *Aircraft Structures for Engineering Students*, Edward Arnold, 1985, p 404.

Vibration of a Disk Backed by an Air-Filled Cylinder

Introduction

The modes of vibration of a thin or thick circular disk are well known, and it is possible to compute the corresponding eigenfrequencies to arbitrary precision from a series solution. The same is true for the acoustic modes of an air-filled cylinder with perfectly rigid walls. A more interesting question is: What happens if the cylinder is sealed in one end not by a rigid wall but by a thin disk?

Model Definition

In COMSOL Multiphysics you can model an air-filled cylinder that is sealed by a thin disk in one end using at least two different approaches. You can describe the pressure in the cavity using an Acoustics application mode, while the model of the disk can be a thin shell in 3D, using shell elements, or a 2D plate. The latter approach is possible thanks to nonlocal couplings and COMSOL Multiphysics' ability to model in different numbers of space dimensions at the same time—*extended multiphysics*.

D. G. Gorman and others have thoroughly investigated the model at hand Ref. 2, and they have developed a semi-analytical solution verified by experiments and simulations. The geometry is a rigid steel cylinder with a height of 255 millimeters and a radius of 38 millimeters. One end is welded to a heavy slab, while the other is sealed with a steel disk only 0.38 millimeters thick. Some of the theoretical eigenfrequencies of a thin disk in vacuum and of a rigidly sealed chamber are given in the following table (according to Ref. 2).

TABLE 4-1: BENCHMARK VALUES FOR EIGENFREQUENCIES OF THE DISK AND THE CYLINDER

NUMBER	CLAMPED DISK IN VACUUM / HZ	RIGIDLY SEALED CYLINDER / HZ
1	671.8	672.5
2	1398	1345
3	2293	2018
4	2615	2645
5	3356	2690
6	4000	4387

Here the coupled system will be modeled using the extended multiphysics approach. This means that the disk is drawn in a 2D geometry and modeled using Mindlin theory DRM plate elements, while the cylinder is drawn in a separate 3D geometry. The acoustics in the cylinder is described in terms of acoustic (differential) pressure. The eigenvalue equation for the pressure is:

$$-\Delta p = \frac{\lambda}{c^2} p$$

The relation between the eigenvalue λ and the eigenfrequency f is $\lambda = (2\pi f)^2$.

A first step is to calculate the eigenfrequencies for the uncoupled disk and cylinder separately and compare them with theoretical values. This way you can verify the model so far and assess the accuracy of the FEM solution.

Results and Discussion

Most of the modes show rather weak coupling between the structural bending of the disk and the pressure field in the cylinder. It is, however, interesting to note that some of the uncoupled modes have been split into one co-vibrating and one contra-vibrating mode with distinct eigenfrequencies. This is the case for modes 1 and 2 and for modes 9 and 12 in the FEM solution. The table below shows a comparison of the eigenfrequencies from the COMSOL Multiphysics analysis with the semi-analytical and experimental frequencies reported by D. G. Gorman and others in Ref. 2. The table also states whether the modes are structurally dominated (str), acoustically dominated (ac), or tightly coupled (str/ac).

TABLE 4-2: RESULTS FROM SEMI-ANALYTICAL AND COMSOL MULTIPHYSICS ANALYSES AND EXPERIMENTAL DATA

TYPE	SEMI-ANALYTICAL / HZ	COMSOL MULTIPHYSICS / HZ	EXPERIMENTAL / HZ
str/ac	636.9	637.2	630
str/ac	707.7	707.7	685
ac	1347	1347.4	1348
str	1394	1395.3	1376
ac	2018	2018.5	2040
str	2289	2293.2	2170
str/ac	2607	2612.1	2596
ac	2645	2645.9	-
str/ac	2697	2696.8	2689

TYPE	SEMI-ANALYTICAL / HZ	COMSOL MULTIPHYSICS / HZ	EXPERIMENTAL / HZ
ac	2730	2730.5	2756
ac	2968	2968.6	2971

The FEM solution is in good agreement with the theoretical as well as the experimental eigenfrequencies. As you might expect from the evaluation of the accuracies for the uncoupled problems, the precision is better for the acoustics-dominated modes.

Modeling Using the Graphical User Interface

MODEL NAVIGATOR

1. Select **2D** in the **Space dimension** list.
2. In the list of application modes, select **Structural Mechanics Module>Mindlin Plate>Eigenfrequency analysis**.
3. Click **OK**.

GEOMETRY MODELING

The geometry of the disk is a solid circle. Its location does not really matter because you will embed into 3D using coupling variables, but the transformations is trivial if you center the disk at the origin:

1. Press the Shift key and click the **Ellipse/Circle (Centered)** button.
2. In the **Circle** dialog box, type 0.038 in the **Radius** edit field and click **OK** to create a circle of radius 0.038 m, centered at the origin.
3. Click the **Zoom Extents** toolbar button.

PHYSICS SETTINGS

Boundary Conditions

The edges of the disk are welded to the cylinder and can therefore be described as rigidly *clamped* or fixed.

1. From the **Physics** menu, choose **Boundary Settings**.
2. Press Ctrl+A to select all boundaries.
3. Make sure that you have selected **Tangent and normal coord. sys (t,n)** in the **Coord. sys.** list.
4. Select **Fixed** in the **Condition** list.

5 Click **OK**.

Subdomain Settings—Material Properties

The steel disk has the following material properties:

- Young's modulus, E=2.1·10^{11}
- Poisson's ratio, ν=0.3
- Density, ρ=7800

1 From the **Physics** menu, choose **Subdomain Settings**.

2 Select subdomain 1.

3 Enter material data according to the following table:

MATERIAL PROPERTY	VALUE
E	2.1e11
ν	0.3
ρ	7800
thickness	0.00038

4 Click **OK**.

MESH GENERATION

To obtain accurate values of the eigenfrequencies of the disk, you need a mesh that is finer than the one produced with the default settings.

1 Open the **Mesh Parameters** dialog box from the **Mesh** menu.

2 Type 0.002 in the **Maximum element size** edit field.

3 Click the **Remesh** button and then click **OK**.

COMPUTING THE SOLUTION

When solving for the eigenfrequencies of the disk in vacuum, only the frequency interval between 500 and 3250 Hz are of interest. Start by searching for the ten first eigenfrequencies (some of these are almost identical and come from double eigenvalues) and make the solver start its search around 500 Hz:

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

2 Type 10 in the **Desired number of eigenfrequencies** edit field.

3 Type 500 in the **Search for eigenvalues around** edit field.

4 Click **OK**.

5 Click the **Solve** toolbar button.

POSTPROCESSING AND VISUALIZATION

1 Click the **3D Surface Plot** button to see the deflection of the disk.

2 From the **Postprocessing** menu, choose **Plot Parameters**.

3 Try looking at some of the eigenmodes by selecting the corresponding eigenfrequencies on the **General** page of the **Plot Parameters** dialog box.

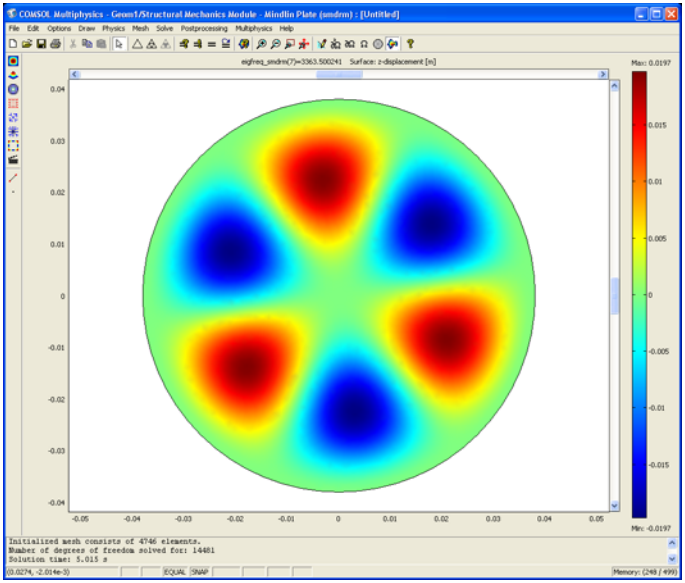


Figure 4-2: The eigenmode associated with the eigenfrequency around 3356 Hz.

You can now compare the eigenfrequencies with the theoretical values for a thin disk. The discrepancy is below 1% for all eigenmodes in the interval, so the conclusion is that the mesh resolution is sufficient.

Adding the 3D Acoustics Application Mode

Now add a second geometry that will contain the cylinder and the acoustic pressure variable using a 3D Acoustics application mode.

1 From the **Multiphysics** menu, choose **Model Navigator**.

2 Click the **Add Geometry** button to add a second geometry.

3 In the **Add Geometry** dialog box, select **3D** in the **Space dimension** list.

4 Click **OK**.

5 In the list of application modes, select **COMSOL Multiphysics>Acoustics>Acoustics>Eigenfrequency analysis**.

6 Click **Add**.

7 Click **OK**.

GEOMETRY MODELING

1 Click the **Cylinder** toolbar button.

2 Type `0.038` in the **Radius** edit field and `0.255` in the **Height** edit field.

3 Click **OK**.

4 Click the **Zoom Extents** button.

PHYSICS SETTINGS

Boundary Conditions

For the moment, all boundaries are assumed to be perfect hard walls.

1 From the **Physics** menu, choose **Boundary Settings**.

2 Press Ctrl+A to select all boundaries.

3 Select the **Sound hard boundary (wall)** button.

4 Click **OK**.

Subdomain Settings

1 From the **Physics** menu, choose **Subdomain Settings**.

2 Select subdomain 1.

3 Type `1.2` in the **Fluid density** (ρ_0) edit field. Leave the sound of speed at its default value.

4 Click **OK**.

MESH GENERATION

Click the **Initialize Mesh** button to create a mesh using the default parameters.

COMPUTING THE SOLUTION

To solve for the acoustic modes only, you must deactivate the Mindlin Plate application mode during the solution. If the plate is not deactivated, COMSOL Multiphysics solves the two eigenvalue problems simultaneously but independently of one another.

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

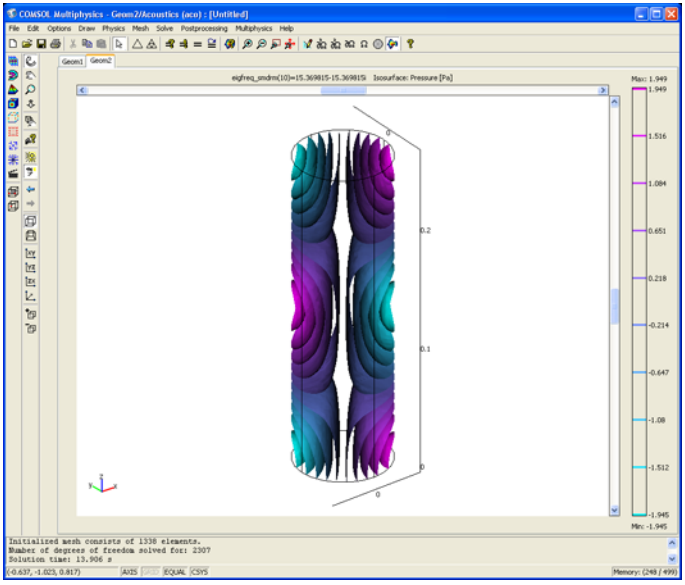
2 Ctrl-click on the **Mindlin Plate (smdrm)** folder to clear that application mode's variables from the list of variables to solve for and then click **OK**.

3 Click the **Solve** toolbar button.

POSTPROCESSING AND VISUALIZATION

1 Clear the **Slice** check box and select the **Isosurface** check box on the **General** page of the **Plot Parameters** dialog box.

2 Try some of the different eigenfrequencies.



You can also compare these eigenfrequencies with the theoretical values in Table 4-1 on page 83. This time, the relative error seems to be one order of magnitude smaller than for the disk, which means that any additional mesh refinement should be done on the plate part.

Coupling the Equations

The first step in the process of coupling the Mindlin plate elements to the acoustic equation is creating the nonlocal couplings. Use these coupling variables to make the pressure available as a load on the plate and the out-of-plane displacement of the plate a valid parameter in the coefficients for the acoustic equation.

OPTIONS AND SETTINGS—COUPLING VARIABLES

First define a coupling variable for the acoustic pressure from the top face of the cylinder to the disk (Mindlin plate):

1. On the **Options** menu, point to **Extrusion Coupling Variables** and then click **Boundary Variables**.
2. In the **Boundary Extrusion Variables** dialog box, select boundary 4 (the top face) of the cylinder.
3. Type `p` in the top row under both **Name** and **Expression**.
4. Click the **Destination** tab.
5. Select **Geom1** in the **Geometry** list and then select subdomain 1 (the disk) in the 2D geometry.
6. Click the **Source Vertices** tab.
7. In the **Vertex selection** list, select three vertices: 2, 4, and 8. Click the **>>** button.
8. Click the **Destination Vertices** tab.
9. In the **Vertex selection** list, select three vertices: 1, 2, and 4. Click the **>>** button.
10. Click **OK**.

Now define a coupling variable for the out-of-plane displacement w from the disk to the top face of the cylinder. Also the time derivative of the displacement, wt, is needed.

1. Switch to the Mindlin Plate application mode by choosing **Geom1: Mindlin Plate (smdrm)** from the **Multiphysics** menu.
2. On the **Options** menu, point to **Extrusion Coupling Variables** and then click **Subdomain Variables**.
3. In the **Subdomain Extrusion Variables** dialog box, select subdomain 1.
4. Type `w` in the top row under both **Name** and **Expression**. Type `wt` in the second row under both **Name** and **Expression**.
5. Click the **Destination** tab.
6. Select `w` from the **Variable** list.
7. Select **Geom2** in the **Geometry** list and then select boundary 4 (the top face).
8. Click the **Source Vertices** tab.
9. In the **Vertex selection** list, select three vertices: 1, 2, and 4. Click the **>>** button.
10. Click the **Destination Vertices** tab.
11. In the **Vertex selection** list, select three vertices: 2, 4, and 8. Click the **>>** button.

12 Click the **Destination** tab.

13 Select `wt` from the **Variable** list.

14 Select **Geom2** in the **Geometry** list and then select boundary 4 (the top face).

15 Click the **Source Vertices** tab.

16 In the **Vertex selection** list, select three vertices: 1, 2, and 4. Click the **>>** button.

17 Click the **Destination Vertices** tab.

18 In the **Vertex selection** list, select three vertices: 2, 4, and 8. Click the **>>** button.

19 Click **OK**.

PHYSICS SETTINGS

Boundary Conditions

The sound-hard boundary condition for a rigid wall is that the normal acceleration vanishes. For a moving wall, such as the thin disk that now seals the cylinder, the condition transforms to

$$\mathbf{n} \cdot \nabla p = -\rho_a a \quad ,$$

where a is the normal acceleration of the wall. As the disk undergoes harmonic motion, its acceleration is proportional to its displacement w. To be more specific: $a = -\lambda w$, where $\lambda = (2\pi f)^2$ is the eigenvalue. Therefore,

$$\mathbf{n} \cdot \nabla p = \rho_a \lambda w$$

You cannot enter contributions to the normal derivative of p that are proportional to the eigenvalue λ directly, because the eigenvalue is not accessible as a variable. Instead you must enter the term λw as `-wt`, where `wt` is the time derivative of w. The Acoustics application mode formulation involves the density in the denominator of the equation, so the right-hand side of the boundary condition is just λw.

1 On the **Physics** menu, point to **Equation System** and then click **Boundary Settings**.

2 Click the **Coefficients** tab.

3 Select the top of the cylinder where the plate is located, that is, boundary number 4.

4 Type `-wt` in the **g** field and click **OK**.

Subdomain Settings

The Mindlin Plate application mode gives first-order eigenvalues. The following steps makes the acoustic equation first order:

1 Select **Equation System** and then **Subdomain Settings** from the **Physics** menu.

2 Copy the expression in the $\mathbf{e}_a$ edit field and paste it into the $\mathbf{d}_a$ edit field.

3 Type `0` in the $\mathbf{e}_a$ edit field.

4 Click **OK**.

The coupling in the other direction, from the air to the disk, is more straightforward:

1 Choose the **Geom1: Mindlin Plate (smdrm)** application mode from the **Multiphysics** menu.

2 Open the **Subdomain Settings** dialog box.

3 Click the **Load** tab.

4 Select subdomain 1.

5 Type `p` in the **Fz** edit field as a surface load on the disk.

6 Click **OK**.

7 Switch back to the 3D geometry by selecting **Geom2: Acoustics (aco)** from the **Multiphysics** menu.

COMPUTING THE SOLUTION

1 Reactivate the Mindlin Plate application mode by selecting all modes in the **Solve for variables** list on the **Solve For** page of the **Solver Manager** dialog box.

2 Click the **Solve** toolbar button to solve the problem.

POSTPROCESSING AND VISUALIZATION

1 Open the **Plot Parameters** dialog box.

2 Add boundary and deformed shape plots in addition to the isosurface plot by selecting the **Boundary** and **Deformed shape** check boxes.

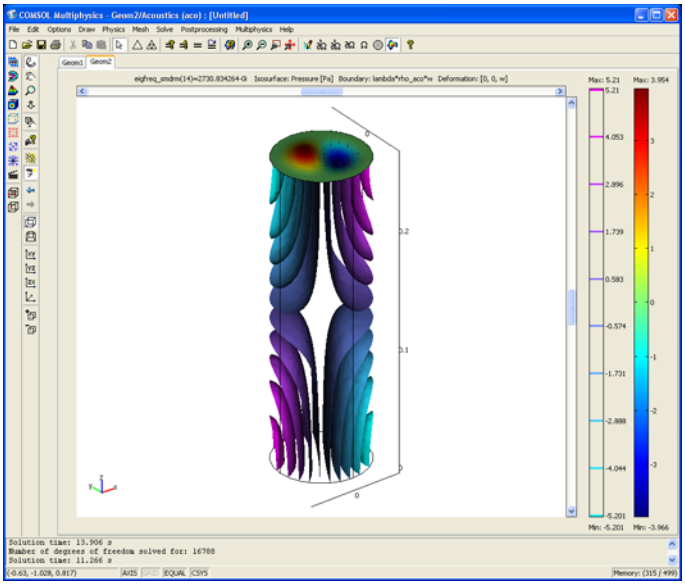
3 Click the **Boundary** tab.

4 Type `lambda*rho_aco*w` in the **Expression** edit field in the **Boundary data** area, that is, the normal acceleration of the disk. On the other boundaries w is not defined so those boundaries will be invisible.

5 Click the **Deform** tab and select the **Boundary** check box only in the **Domain types to deform** area.

6 In the **Deform data** area, click the **Boundary** tab and type `0`, `0`, and `w` in the **x component**, **y component**, and **z component** edit field, respectively.

7 Click **Apply** to plot the solution. Examine the different eigenmodes.



Reference

2. D. G. Gorman, J. M. Reese, J. Horacek, and D. Dedouch: "Vibration analysis of a circular disk backed by a cylindrical cavity," *Proc. Instn. Mech. Engrs.*, vol. **215**, Part C, 2001.

Stresses in the Soil Surrounding a Traffic Tunnel

Introduction

This model demonstrates the use of elastoplastic analysis in soil mechanics using the Mohr-Coulomb material model under plane strain conditions. The analysis studies the stresses and deformations in the soil surrounding a traffic tunnel. This model is inspired by a model in Ref. 3.

Model Definition

The modeling of the tunnel and the surrounding soil consist of the following stages:

1. Excavation of the soil at the site.
2. Adding the tunnel onto the soil.
3. The top of the tunnel and the soil surface are subjected to pressure.

Due to symmetry reason the model only includes one half of the geometry (see Figure 4-3).

A suitable 2D application mode to model geotechnical problems as this is the Plane Strain application mode. To model the above stages in the Structural Mechanics Module, you use a model with two plane strain application modes.

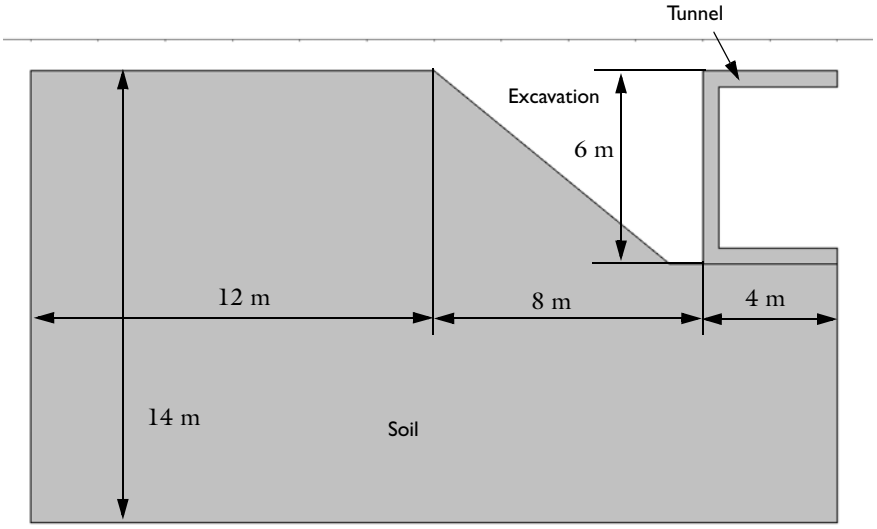


Figure 4-3: Cross section of the tunnel and the soil layer.

MATERIAL

Soil is a material with a highly nonlinear stress-strain behavior. The soil properties can be divided into strength and stiffness properties. The strength of soil can be characterized by the effective strength parameters c (cohesion) and ϕ (friction angle). The stiffness parameters E (Young's modulus) and ν (Poisson's ratio) describes the elastic deformation of the soil.

Problems of material failure in soil mechanics and other frictional materials such as concrete are often modeled using the well-known Mohr-Coulomb law. This is an elastic perfectly-plastic material model. A smooth approximation of the Mohr-Coulomb yield surface for a three-dimensional case is the Drucker-Prager model.

The failure criteria, the yield surface, for both material models are based on the two strength parameters, cohesion c and friction angle ϕ, and two stress quantities, the hydrostatic stress and the equivalent deviatoric stress.

The yield surface F for the Drucker-Prager material law is given by:

$$F = 3 \cdot \alpha \cdot \sigma_m + \sigma_{eqv} - K$$

where

$$\alpha = \frac{2 \cdot \sin\phi}{\sqrt{3} \cdot (3 - \sin\phi)}$$

and

$$K = \frac{6 \cdot c \cdot \cos\phi}{\sqrt{3} \cdot (3 - \sin\phi)}$$

where σ_m is the hydrostatic stress (or mean stress), and σ_{eqv} is the equivalent deviatoric stress.

On the other hand, if two-dimensional plane strain conditions prevails the Drucker-Prager criterion becomes identical to the Mohr-Coulomb criterion if you write the above material parameters as (see Ref. 5):

$$\alpha = \frac{\tan\phi}{\sqrt{(9 + 12 \cdot \tan^2\phi)}}$$

$$K = \frac{3 \cdot c}{\sqrt{(9 + 12 \cdot \tan^2\phi)}}$$

The hydrostatic stress is defined using the normal stress components σ_{ii} by (see Ref. 4):

$$\sigma_m = \frac{\sigma_{ii}}{3} = \frac{\sigma_x + \sigma_y + \sigma_z}{3}$$

whereas the equivalent deviatoric stress is defined using the deviatoric stress components s_{ij}:

$$\sigma_{eqv} = \sqrt{\frac{1}{2} \cdot s_{ij} \cdot s_{ij}} = \sqrt{\frac{1}{2} \cdot (s_x^2 + s_y^2 + s_z^2) + s_{xy}^2 + s_{yz}^2 + s_{zx}^2}$$

where the deviatoric stress components s_{ij} are defined by:

$$s_{ij} = \sigma_{ij} - \frac{\delta_{ij} \cdot \sigma_{kk}}{3} \qquad \text{or}$$

$$s_x = \sigma_x - \sigma_m \qquad s_{xy} = \tau_{xy}$$
$$s_y = \sigma_y - \sigma_m \qquad s_{yz} = \tau_{yz}$$
$$s_z = \sigma_z - \sigma_m \qquad s_{zx} = \tau_{zx}$$

The material properties for the soil are tabulated in Table 4-3 below.:

TABLE 4-3: MATERIAL PROPERTIES FOR THE SOIL.

QUANTITY	NAME	EXPRESSION	UNIT
Young's modulus	E	10e6	Pa
Poisson's ratio	ν	0.3	-
Cohesion	c	10000	Pa
Friction angle	ϕ	35	deg
Specific weight	γ	18000	N/m^3

The model assumes that the tunnel behaves completely elastically and uses some typical material values for concrete:

TABLE 4-4: MATERIAL PROPERTIES FOR THE TUNNEL.

QUANTITY	NAME	EXPRESSION	UNIT
Young's modulus	E	25e9	Pa
Poisson's ratio	ν	0.33	-
Specific weight	γ	25000	N/m^3

CONSTRAINTS AND LOADS

- The lower horizontal boundary is restrained from moving in both the x- and y-directions, thereby simulating a rough and rigid underlying rock layer.
- The model only includes one half of the tunnel due to symmetry. The displacements in the normal direction (x-direction in this case) are constrained on the symmetry-cut boundaries.
- The left vertical boundary is assumed to be perfectly smooth and rigid. This is modeled by applying a constraint only in the horizontal direction while allowing movement in the vertical direction (see also Figure 4-4).
- The specific weight of the soil is entered as a domain load in the negative y-direction. The solver ramps up the specific weight and the in-situ stress, which you enter as an initial stress, during the elastic-perfectly plastic analysis in the first excavation stage of the analysis (solver parameter values `para` from 0 to 1). Originally, the soil carries the following in-situ stresses before the excavation:

$$\sigma_x = \lambda \cdot \sigma_y$$
$$\sigma_y = -\gamma \cdot y$$
$$\sigma_z = \lambda \cdot \sigma_y$$
$$\lambda = \frac{\nu}{(1-\nu)}$$

where γ denotes the specific weight of the soil, y is the vertical coordinate from the ground level, and ν is Poisson's ratio.

- The specific weight of the tunnel is ramped up and applied as a domain load in the negative y-direction in the second stage of the model (solver parameter values `para` from 1 to 2).
- Additional pressure loads are also applied in the second stage. The first pressure load is ramped up to $5 \cdot 10^4$ Pa on the upper horizontal boundaries of the tunnel domain and a second pressure load is ramped up to $15 \cdot 10^4$ Pa on the inner lower horizontal boundaries.
- The vertical deformation of the lower horizontal edge of the tunnel, which is the common boundary with the soil domain, is set to be equal with the vertical deformation of the soil.

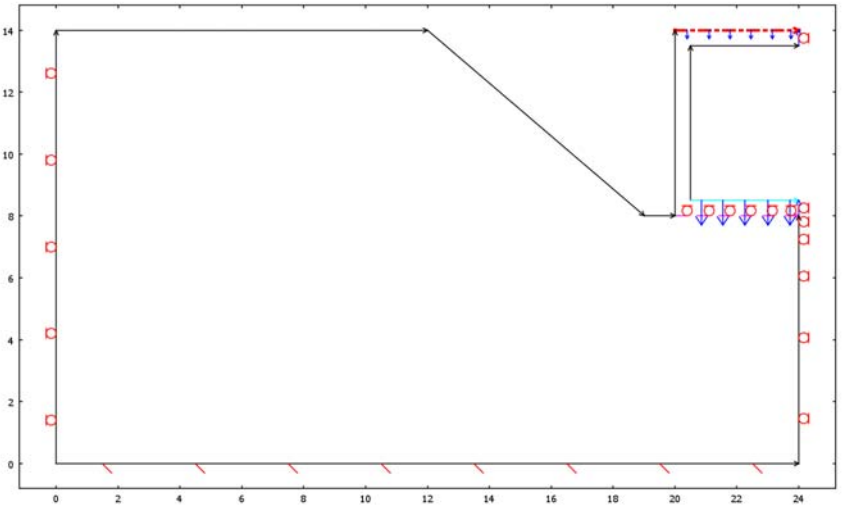


Figure 4-4: Boundary constraints and loads.

Modeling in COMSOL Multiphysics

The model includes two Plane Strain application modes. The first application mode is active in the soil domain and computes the elastic perfectly-plastic response. The response due to the excavation and the in-situ stress is calculated in the first stage of the model, where the specific weight of the soil and the in-situ stresses are ramped up using the parametric solver for parameter values between 0 and 1.

The second stage computes the soil response due to the additional weight from the tunnel and boundary loads on the tunnel. In this analysis the parametric solver ramps up the weight and the loads in the same way as in the first stage but with parameter values from 1 to 2.

The second application mode is active in the tunnel domain and models the elastic deformations of the tunnel construction. The use of a separate application mode for the tunnel domain makes it possible to apply a tangential slip conditions between the soil domain and the tunnel domain.

The implemented elastic perfectly-plastic material model is the Mohr-Coulomb material model using a user-defined yield function and a perfectly plastic hardening function. The yield function contains hydrostatic and equivalent deviatoric stresses as well as the material property constants α and K. Both the stress variables and the material properties are defined as scalar expressions. A second set of scalar expressions of these variables are also defined using variables evaluated at the Gauss points. The second set of variables are for postprocessing purposes.

Convergence problems can occur in elastoplastic problems like this. A couple of remedies to these problems are to use the mixed formulation and to manually tune the parameter step-size settings. In this model you specify the initial parameter step and the minimum step size, while keeping the default setting for the maximum step size.

Sometimes the internal scaling of the solution variables can also cause problems. The remedy to this problem is to turn off the automatic scaling or use the initial value scaling.

Results and Discussion

You can study the development of the stress levels, plastic regions, and deformations in the following figures. Figure 4-5 shows the stress levels and the plastic regions after the excavation. The red colored areas indicate the plastic regions, and a contour plot shows the equivalent deviatoric stresses.

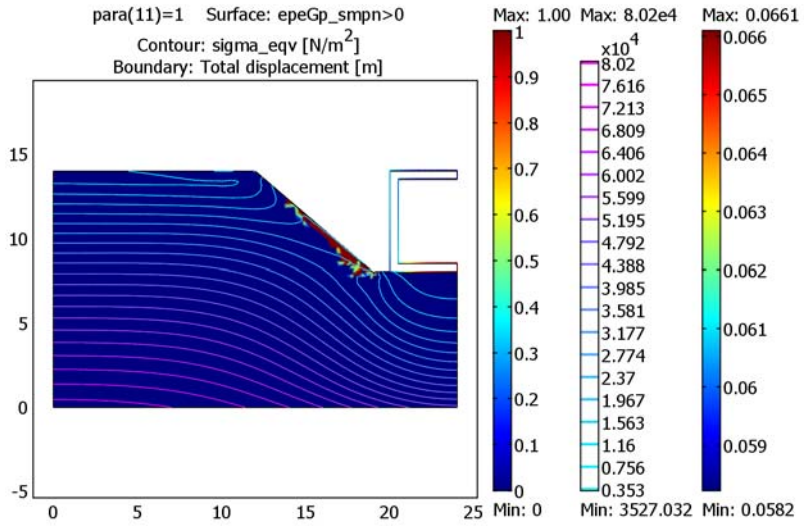


Figure 4-5: The equivalent deviatoric stresses and plastic regions due to the excavation.

Figure 4-6shows the equivalent deviatoric stresses in the soil domain as well as the deformations when the surface pressures and the specific weight of the tunnel are applied. The stresses has reached the yield surface in the red colored areas.

The figure also shows that the maximum final vertical deformations of the tunnel construction are approximately 5 cm.

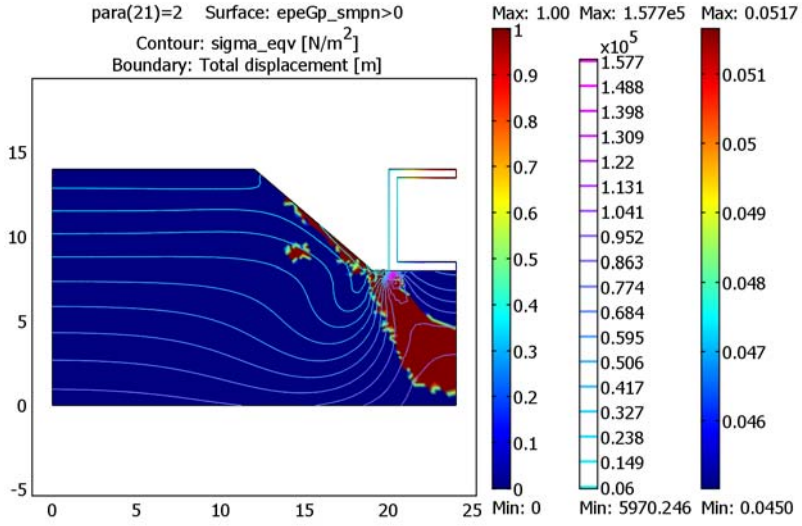


Figure 4-6: The equivalent deviatoric stresses and deformations at peak surface pressure.

References

3. I. Doltsinis., *Elements of Plasticity*, WIT Press, 2000.

4. O.C. Zienkiewicz, *The Finite Element Method*, McGraw-Hill, 1991.

5. W.F. Chen, *Nonlinear Analysis in Soil Mechanics*, Elsevier, 1990

Modeling Using the Graphical User Interface

MODEL NAVIGATOR

1 Select **2D** in the **Space dimensions** list on the **New** page in the **Model Navigator.**

2 Select **Structural Mechanics Module>Plane Strain>Static analysis elasto-plastic material**.

3 Click the **OK** button.

GEOMETRY MODELING

1 Add a rectangle by selecting from the **Draw** menu, **Specify Object**, and then click **Rectangle**.

2 Enter 24 into the **Width** edit field and 14 into the **Height** edit field.

3 Click **OK**.

4 Next specify some lines. On the **Draw** menu, point to **Specify Object**, and then click **Line**.

5 Enter into the **x** edit field and into the **y** edit field the following coordinate pairs for each line and click **OK** between each line.

LINE	X	Y
1	12 19	14 8
2	19 24	8 8

6 Select all geometry objects by pressing Ctrl+A.

7 Click the **Coerce to Solid** toolbar button.

8 Click the **Split Object** button.

9 Select the geometry object C03 by clicking on it and delete it by pressing **Delete**.

10 Add a rectangle by selecting from the **Draw** menu, **Specify Object**, and then click **Rectangle**.

11 Enter 4 into the **Width** edit field and 6 into the **Height** edit field.

12 Enter also 20 into the **x** edit field and 8 into the **y** edit field in the **Position** square.

13 Click **OK**.

14 Add a second rectangle by selecting from the **Draw** menu, **Specify Object**, and then click **Rectangle.**

15 Enter 3.5 into the **Width** edit field and 5 into the **Height** edit field.

16 Enter also 20.5 into the **x** edit field and 8.5 into the **y** edit field in the **Position** area.

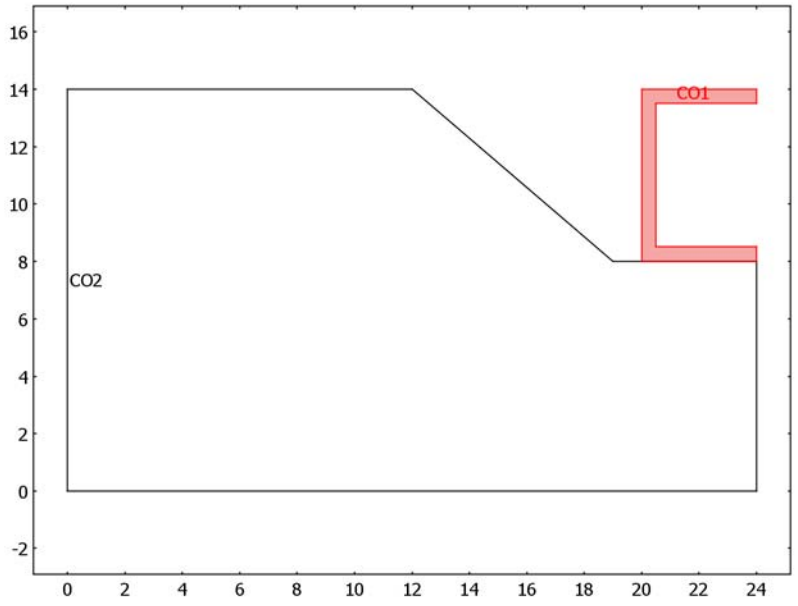
17 Click **OK**.

18 Click the **Create Composite Object** button.

19 Write R1-R2 in the **Set Formula** edit field.

20 Click **OK**.

21 Click **Zoom Extents**.



OPTIONS AND SETTINGS

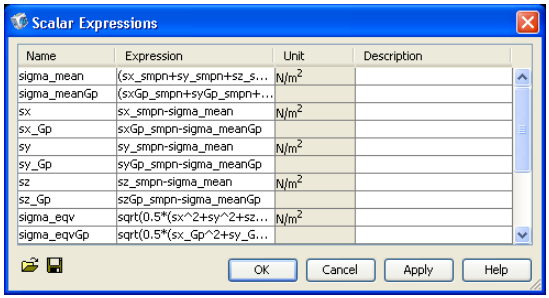
Scalar Expression

1 From the **Options** menu, choose **Expression** and **Scalar Expressions**.

2 Specify expressions according to the following table:

NAME	EXPRESSION
sigma_mean	(sx_smpn+sy_smpn+sz_smpn)/3
sigma_meanGp	(sxGp_smpn+syGp_smpn+szGp_smpn)/3
sx	sx_smpn-sigma_mean
sx_Gp	sxGp_smpn-sigma_meanGp
sy	sy_smpn-sigma_mean
sy_Gp	syGp_smpn-sigma_meanGp
sz	sz_smpn-sigma_mean
sz_Gp	szGp_smpn-sigma_meanGp
sigma_eqv	sqrt(0.5*(sx^2+sy^2+sz^2)+sxy_smpn^2)

NAME	EXPRESSION
sigma_eqvGp	sqrt(0.5*(sx_Gp^2+sy_Gp^2+sz_Gp^2)+sxyGp_smpn^2)
alpha	tan(phi_r)/sqrt(9+12*(tan(phi_r))^2)
K	(3*c)/sqrt(9+12*(tan(phi_r))^2)
F	3*alpha*sigma_mean+sigma_eqv-K

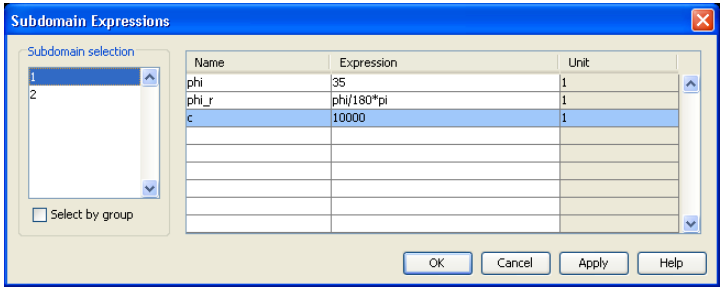


Subdomain Expressions

1 From the **Options** menu, choose **Expression>Subdomain Expressions**.

2 Specify expressions according to the following table for subdomain 1:

NAME	EXPRESSION
phi	35
phi_r	phi/180*pi
c	10000



PHYSICS SETTINGS

Boundary Settings

1 Select **Boundary Settings** from the **Physics** menu.

2 Specify boundary settings according to the following table:

BOUNDARY	1, 12		2	
Page	Constraint		Constraint	
	R_x	0	R_x	0
			R_y	0

Subdomain Settings

1 Select **Subdomain Settings** from the **Physics** menu.

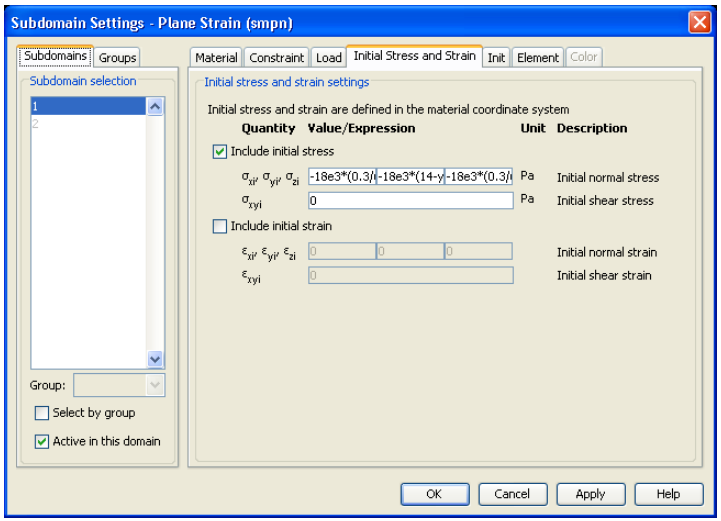
2 Select subdomain 2 in the **Subdomain selection** list.

3 Clear the **Active in this domain** check box to deactivate this subdomain in this application mode.

4 Select subdomain 1 in the **Subdomain selection** list.

5 Enter subdomain data according to the following table:

SUBDOMAIN	1	
Page	Material	
	Material model	Elasto-plastic material
	E	10e6
	ν	0.3
Page	Load	
	F_y	-18e3*para*(para<=1)-18e3*(para>1)
Page	Initial Stress and Strain	
	σ_{xi}	-18e3*(0.3/(1-0.3))*(14-y)*para*(para<=1)-18e3*(0.3/(1-0.3))*(14-y)*(para>1)
	σ_{yi}	-18e3*(14-y)*para*(para<=1)-18e3*(14-y)*(para>1)
	σ_{zi}	-18e3*(0.3/(1-0.3))*(14-y)*para*(para<=1)-18e3*(0.3/(1-0.3))*(14-y)*(para>1)
	σ_{xyi}	0



6 Select the **Use mixed U-P formulation (nearly incompressible material)** check box on the **Material** page.

7 Click the **Elasto-plastic material data** button.

8 Select **Perfectly plastic** from the **Hardening model** list in the **Elasto-plastic material settings** dialog box.

9 Select **User defined** from the **Yield function** list and type `F` into the **Yield function** edit field.

10 Type `0` into the **Yield stress** level edit field.

11 Click **OK** to close the **Elasto-plastic material settings** dialog box.

12 Click **OK to** close the **Subdomain Settings** dialog box.

MESH GENERATION

1 Click the **Initialize Mesh** toolbar button to generate the mesh.

2 Click the **Refine Mesh** toolbar button to refine the mesh.

MODEL NAVIGATOR

Next add a second Plane Strain application mode:

1 Select **Model Navigator** from the **Multiphysics** menu.

2 Select **2D** in the **Space dimension** list on the **New** page in the **Model Navigator.**

3 Select **Structural Mechanics Module>Plane Strain>Static analysis**.

4 Click **OK**.

PHYSICS SETTINGS

Subdomain Settings

1 Select **Subdomain Settings** from the **Physics** menu.

2 Select subdomain 1 in the **Subdomain selection** list.

3 Clear the **Active in this domain** check box to deactivate this subdomain in this application.

4 Enter subdomain data according to the following table.

SUBDOMAIN	2	
Page	Material	
	Material model	Isotropic material
	E	`25e9*(para-1)*(para>1)+100`
	ν	`0.33`
Page	Load	
	Fy	`-25e3*(para-1)*(para>1)`

5 Click **OK** to close the **Subdomain Settings** dialog box.

Boundary Settings

1 Select **Boundary Settings** from the **Physics** menu.

2 Specify boundary settings according to the following tables:

BOUNDARY	13, 14	
Page	Constraint	
	R_x	0

BOUNDARY	7	
Page	Constraint	
	R_y	v

BOUNDARY	8	
Page	Load	
	F_y	`-5e4*(para>1)*(para-1)`

BOUNDARY	10	
Page	Load	
	F_y	`-15e4*(para>1)*(para-1)`

COMPUTING THE SOLUTION

1 Select **Solver Parameters** from the **Solve** menu.

2 Type the name `para` in the **Name of parameter** edit field.

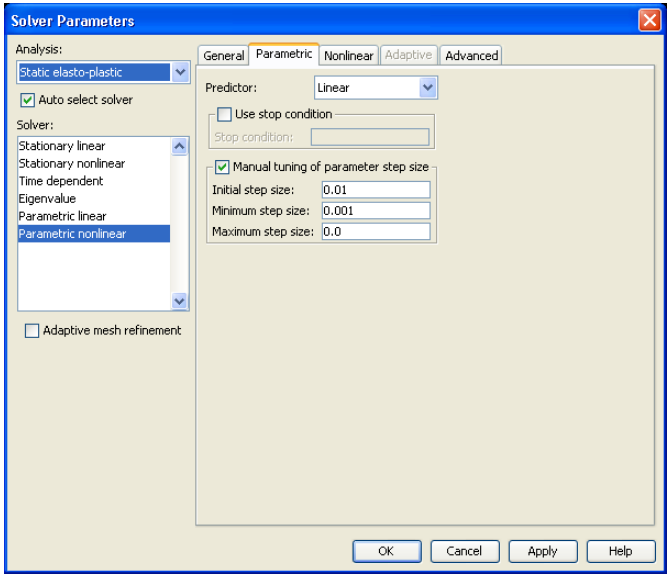
3 Enter `0:0.1:2` in the **List of parameter values** edit field.

4 Click the **Parametric** tab.

5 Select **Manual tuning of parameter step size**.

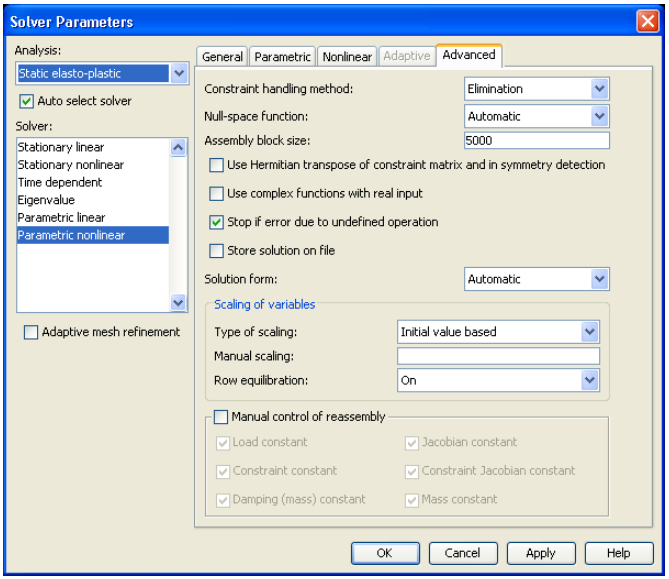
6 Type `0.01` into the **Initial step size** edit field.

7 Type `0.001` into the **Minimum step size** edit field.



8 Click the **Advanced** tab.

9 Select **Initial value based** from the **Type of scaling** list.



10 Click **OK** to close the **Solver Parameters** dialog box.

11 Click the **Solve** toolbar button.

POSTPROCESSING AND VISUALIZATION

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

2 Select 1 from the **Parameter value** list.

3 Select the **Surface** page.

4 Type `epeGp_smpn>0` in the **Expression** edit field to plot the areas where plastic deformation has occurred.

5 Select the **Contour** page and select the **Contour plot** check box.

6 Type `sigma_eqv` in the **Expression** edit field to plot the equivalent deviatoric stress.

7 Click on the **Boundary** tab and select the **Boundary plot** check box.

8 Select the **Total displacement (smpn2)** from the **Predefined quantities** list.

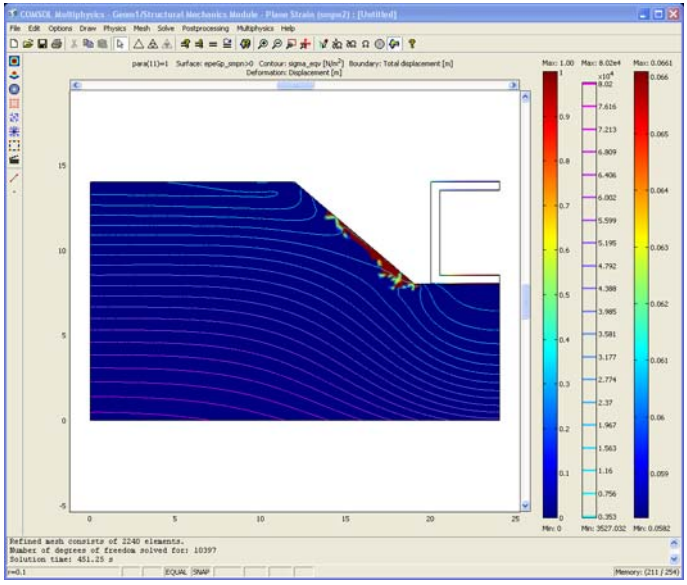
9 Click the **Deform** tab and select the **Deform plot** check box.

10 Select the **Displacement (smpn)** from the **Predefined quantities** list in the **Subdomain Data** tab and **Displacement (smpn2)** from the **Predefined quantities** list in the **Boundary Data** tab.

11 Clear the **Scale factor** check box.

I2 Type 1 into the **Scale factor** edit field.

I3 Click **OK**.



Model Documentation Updates

Obstacle in Fluid Model

In the section "Boundary Conditions," the table with boundary settings in Step 5 on page 217 in the *Structural Mechanics Module Model Library* should say `-lm7`, `-lm8`, and `-lm9` instead of `-lm10`, `-lm11`, and `-lm12` for the force components F_x, F_y, and F_z, respectively. These force components define the loads on all boundaries except boundary 8.

5

MEMS Module 3.2b Model Library Update

This chapter contains the following new models for the MEMS Module 3.2b Model Library:

- Thermoelastic damping: 2D and 3D models
- Microchannel H-cell
- Electroosmotic biochip

Thermoelastic Damping in a MEMS Resonator

Introduction

A high Q-value is a key factor of a MEMS resonator. It it essential that the resonator vibrates consistently at the desired frequency and that it requires as little energy as possible to maintain its vibration. These features can be characterized by the Q-value of the resonator, which is a measure of the sharpness of the resonator spectrum's peak. There are several ways to define the Q-value, for example:

$$Q = \frac{2\pi W_0}{\Delta W} = \frac{\omega_0}{2\delta} = \frac{\omega_0}{\Delta\omega} \tag{5-1}$$

where W_0 is the total stored vibrational energy, ΔW is the energy lost per cycle, ω_0 is the natural angular frequency, δ is the damping factor (vibration decays exponentially with δt), and $\Delta\omega$ is the half power width of the spectrum.

In order to improve the resonator, the designer needs to consider all aspect that produce damping and noise to the system. For example, resonators are usually run in vacuum to minimize effects of air and squeeze film damping.

Thermoelastic damping (Ref. 1, Ref. 2, and Ref. 3) is an important factor that the resonator designer needs to address. It is a result from phenomena called thermoelastic friction, which takes place when you subject any material to cyclic stress. The stress results in deformation, and the required energy is mostly stored as internal potential energy. However, it is known that material heats under compressive stress and cools under tensile stress. Thus, due to the heat flow from warmer to cooler regions energy is also lost as nonrecoverable thermal energy. The amount of thermoelastic friction and damping depends on the rate of this energy loss. The magnitude of the energy loss depends on the vibrational frequency and on the structure's thermal relaxation time constant, which is the effective time the material requires to relax after an applied constant stress or strain. Therefore, the effect of thermoelastic dissipation, and consequently the damping, is most pronounced when the vibration frequency is close to the thermal relaxation frequency.

For simple structures, researchers have developed analytical expressions to estimate thermoelastic damping. According to Zener (Ref. 1 and Ref. 2), you can calculate the Q-value for a resonator with a single thermal mode by:

$$\frac{1}{Q} = \left(\frac{E\alpha^2 T_0}{\rho C_p}\right)\left(\frac{\omega\tau}{1+(\omega\tau)^2}\right) \tag{5-2}$$

where E is the Young's modulus, α is the thermal expansion coefficient, T_0 is the resonator temperature at rest, ρ is the density, C_p is the heat capacity of the material, ω is the vibration angular frequency, and τ is the thermal relaxation time of the system. Thus it is easy to see that in order to have good Q-value, the system needs to be designed so that ω is as far from $1/\tau$ as possible.

The natural frequency of a beam clamped at both ends can be calculated by (Ref. 4):

$$\omega_0 = a_0^2 \frac{h}{L^2}\sqrt{\frac{E}{12\rho}} \tag{5-3}$$

where a_0 equals 4.730, h and L are the thickness and length of the beam, and E and ρ are material parameters as above.

Thermal relaxation time of the beam is given by

$$\tau = \frac{\rho C_P h^2}{\pi^2 \kappa} \tag{5-4}$$

where κ is thermal conductivity and other parameters are as above.

The problem is that equations are valid only for very simple structures. Therefore more advanced methods, such as FEM, are preferable.

This example shows how to model thermoelastic damping with COMSOL Multiphysics. To be able to compare with measurements and analytical expressions, this example illustrates a simple beam resonator in 2D and 3D. The Q value and natural frequency is solved with an eigenfrequency analysis that combines heat transfer and structural mechanics in one equation system. Thus the eigenmodes are thermoelastic.

This example was inspired by the work of Amy Duwel and others (Ref. 1).

Model Definition

The Figure 5-1shows the geometry. The resonator is a beam of silicon with length 400 µm, height 12 µm, and width 20 µm. The beam is fixed at both ends, and it vibrates in a flexural mode in the *z*-direction (that is, along the smallest dimension). The model assume that the vibration takes place in vacuum. Thus, there is no transfer of heat from the free boundaries. The model also assumes that the contact boundaries are thermally insulated.

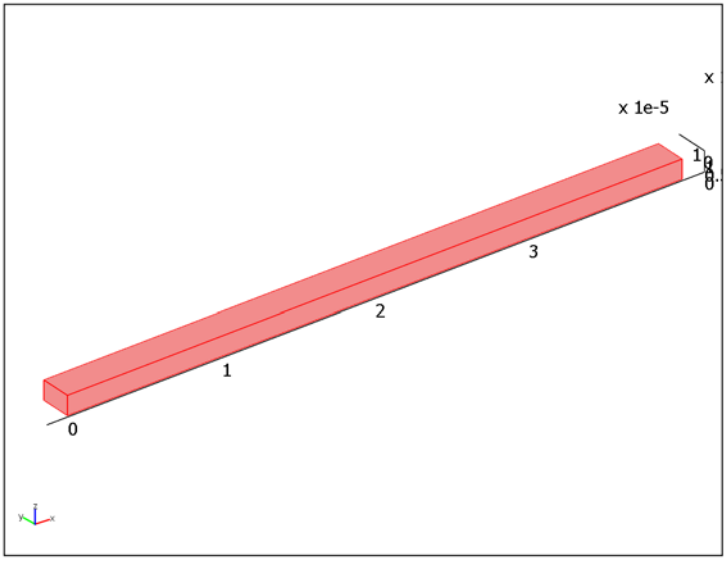


Figure 5-1: Geometry of the modeled beam. The beam is fixed at both ends.

Table 5-1 lists the physical properties of the beam material and the surroundings:

TABLE 5-1: MATERIAL PROPERTIES (POLYSILICON)

PROPERTY	VALUE
Young's modulus, E	157 GPa
Density, ρ	2330 kg m^{-3}
Poisson's ratio, ν	0.3
Thermal expansion coefficient, α	2.6e-6 K^{-1}
Specific Heat, C_P	700 J kg^{-1} K^{-1}

TABLE 5-1: MATERIAL PROPERTIES (POLYSILICON)

PROPERTY	VALUE
Thermal Conductivity, k	90 W m^{-1} K^{-1}
Ambient and initial beam temperature, T_{init}	300 K

To gain information about the quality of the resonator, it is of interest to know its natural frequency and Q-value. To do this, you run an eigenfrequency analysis to find the eigenvalues for this system. For a system with damping, the eigenvalue λ contains information about the natural frequency and Q-value according to:

$$w_0 = |\text{imag}(\lambda)|$$
$$Q = \left|\frac{\text{imag}(\lambda)}{2\,\text{real}(\lambda)}\right| \tag{5-5}$$

The eigenvalues appear as complex conjugates, and ω_0 and Q are therefore given as absolute values.

At this point, to avoid any confusion, it is good to note that here Q refers to the quality of the resonator, whereas later in this text Q_{heat} refers to the heat source term in the heat equation.

To model thermoelastic damping, you must consider both the thermal problem and the structural problem. Furthermore, there is a two-way coupling between the two: The strain rate heats or cools the material locally, which produces thermal strains.

The relation between the material stress σ and strain ε is given by

$$\sigma = D\varepsilon_{el} = D(\varepsilon - \varepsilon_{th}) \tag{5-6}$$

where ε_{el} and ε_{th} are the elastic and thermal strains, respectively, D is the 6x6 elasticity matrix, and all stresses and strains are denoted with 6-component vectors consisting of x, y, and z normal components followed by the xy, yx, and xz shear components.

It is possible to expand this for an isotropic material:

$$\begin{bmatrix}\sigma_x\\ \sigma_y\\ \sigma_z\\ \sigma_{xy}\\ \sigma_{yz}\\ \sigma_{xz}\end{bmatrix} = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix}1-\nu & \nu & \nu & 0 & 0 & 0\\ \nu & 1-\nu & \nu & 0 & 0 & 0\\ \nu & \nu & 1-\nu & 0 & 0 & 0\\ 0 & 0 & 0 & \frac{1-2\nu}{2} & 0 & 0\\ 0 & 0 & 0 & 0 & \frac{1-2\nu}{2} & 0\\ 0 & 0 & 0 & 0 & 0 & \frac{1-2\nu}{2}\end{bmatrix}\begin{bmatrix}\varepsilon_x-\alpha(T-T_{ref})\\ \varepsilon_y-\alpha(T-T_{ref})\\ \varepsilon_z-\alpha(T-T_{ref})\\ \varepsilon_{xy}\\ \varepsilon_{yz}\\ \varepsilon_{xz}\end{bmatrix} \tag{5-7}$$

where T is the strain temperature, T_{ref} is the stress free reference temperature, and α is the thermal expansion coefficient.

The eigenfrequency analysis is based on the transient equation for force equilibrium:

$$\rho\frac{\partial^2\mathbf{u}}{\partial t^2} = \nabla\cdot\sigma \tag{5-8}$$

where ρ is the material density, $\mathbf{u}$ is the deformation vector, and σ is now given as the symmetric stress tensor. COMSOL Multiphysics translates this to an eigenfrequency problem with replacement $\partial\mathbf{u}/\partial t = -\lambda\mathbf{u}$ and $\partial^2\mathbf{u}/\partial t^2 = \lambda^2\mathbf{u}$, where λ is the eigenvalue.

You solve the thermal problem within the material with the heat equation

$$\rho C_P\frac{\partial T}{\partial t} - \nabla\cdot(k\nabla T) = Q_{heat} \tag{5-9}$$

where ρ is the material density, C_P is the material heat capacity, k is the thermal conductivity, and Q_{heat} it the heat source. In this example, Q_{heat} represents the thermoelastic heating, which for an isotropic material has the form

$$Q_{heat} = -\frac{E\alpha T_{init}}{1-2\nu}\frac{\partial(\varepsilon_x+\varepsilon_y+\varepsilon_z)}{\partial t} \tag{5-10}$$

where E, α, and ν are the same as in structural problem, and T_{init} is the initial (stress free) temperature. ε_x, ε_y, and ε_z are the normal strain components.

When modeling in 2D, the model uses the plane stress approximation, so the third stress component vanishes. Thus, from Equation 5-7 you get

$$\sigma_z = \frac{E}{(1+\nu)(1-2\nu)}(\nu\varepsilon_x + \nu\varepsilon_y + (1-\nu)\varepsilon_z - \alpha(T - T_{ref})(1+\nu)) = 0 \quad (5\text{-}11)$$

and

$$\varepsilon_z = \frac{-\nu}{1-\nu}(\varepsilon_x + \varepsilon_y) + \frac{1+\nu}{1-\nu}\alpha(T - T_{ref}) \quad (5\text{-}12)$$

Substituting this into Equation 5-10 (and changing T_{ref} to T_{init}), the heat source for plane stress becomes

$$Q_{heat} = -\frac{E\alpha T_{init}}{(1-2\nu)(1-\nu)}\left[(1-2\nu)\frac{\partial(\varepsilon_x + \varepsilon_y)}{\partial t} + (1+\nu)\alpha\frac{\partial T}{\partial t}\right] \quad (5\text{-}13)$$

Results and Discussion

Figure 5-2 and Figure 5-3 show the eigenmodes and temperature distribution corresponding to the found eigenvalue. Solved natural frequencies and Q-values are given in Table 5-2. Reference data, one calculated with Equation 5-2 to Equation 5-4 and the other from measurements (see Ref. 1) are also given.

TABLE 5-2: COMPARISON OF NATURAL FREQUENCY AN Q-VALUE

SOURCE	F0 (MHZ)	Q
3D model	0.63 MHz	9160
2D model	0.63 MHz	10190
Equation	0.63 MHz	10260
Measurements (Ref. 1)	0.57 MHz	10281

The Q-value given by the 3D model appear to be roughly 10% smaller than the other estimates. One reason for this difference comes from the simplifying assumption that concerns both the Zener's equation (Equation 5-2) and the plane stress method. For example, the plane stress method assumes that the structure is very thin and that there are no stresses perpendicular to the plane. However, looking at the 3D model in more detail, you find that stresses and their spatial derivatives have components of equal magnitude in all three dimensions. The model also assumes that the beam was perfectly fixed at its ends. Physically, this cannot happen, and allowing more loose contact lowers the natural frequency and improves the Q-value.

One factor that also affects the simulated Q-values is the boundary condition for the thermal equation. This example uses thermal insulation on all boundaries, but you can assume that there is a flux of heat at least from both ends of the beam. If you simply

assign constant temperature (T = 0) to the beam ends, the Q-value improves considerably. The physically correct result should be somewhere between these two estimates.

A look at Figure 5-2 and Figure 5-3 shows a temperature distribution that appears to be correct according to the theory: There is an increased temperature near the compressive strain and a decreased temperature near the tensile strain. However, you should not take the temperature amplitude (–4 to 4 K) literally, because the software normalizes the solution from the eigenfrequency solver.

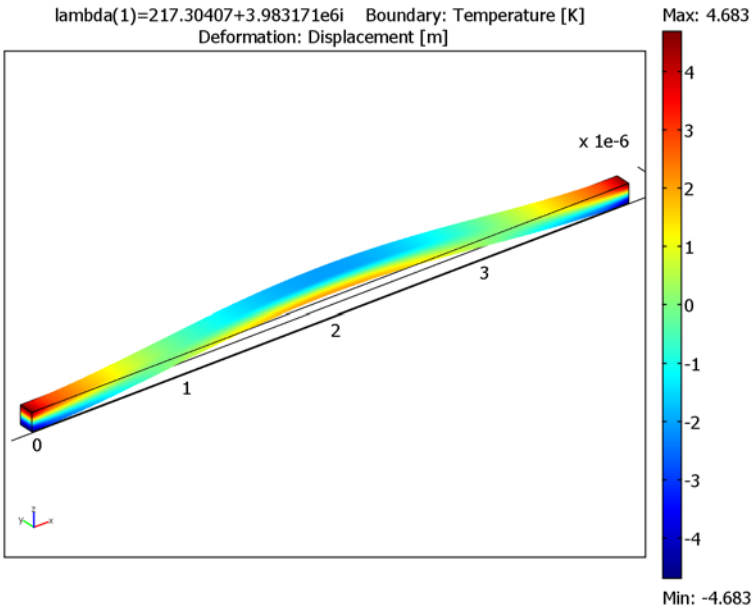


Figure 5-2: First eigenmode and temperature distribution of the 3D model.

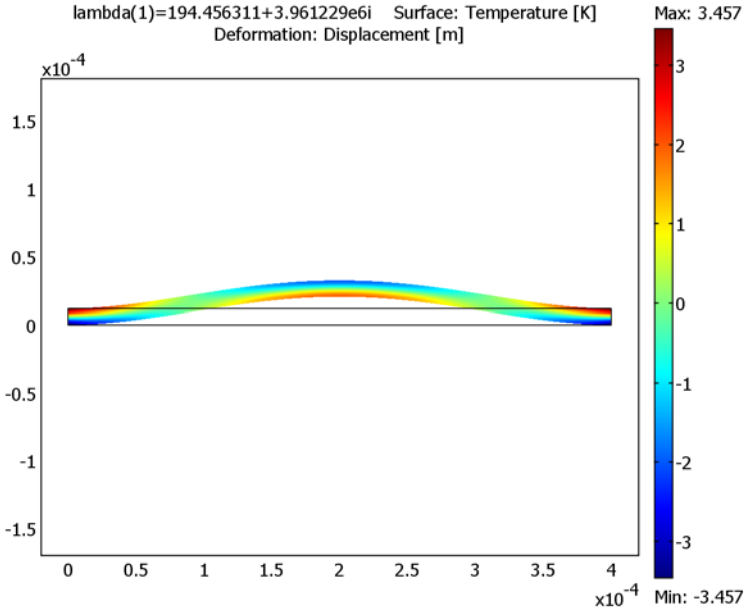


Figure 5-3: First eigenmode and temperature distribution of the 2D model.

Modeling in COMSOL Multiphysics

To create a model of thermoelastic damping with COMSOL Multiphysics requires two application modes. For the 3D model, use the Solid, Stress-Strain application mode from the MEMS Module and the Conduction application mode from the base package. In 2D, use the Plane Stress application mode from the MEMS Module with the same Conduction application mode as in 3D.

The eigenfrequency analysis in the structural mechanics application modes normally uses only the second time derivative of the deformation variables. However, the thermoelastic equation system also contains first time derivatives of the temperature T and the deformation gradients, which are essential for the solution. Therefore, to include all time derivatives in the equations, you initialize all application modes using their transient formulations and use the solver in the transient analysis mode. Then you switch to the eigenfrequency solver. This way, the eigenfrequency solver finds all time derivatives and replaces them with the correct eigenvalue expressions, and the solution contains all information of the model.

In this example you guide the eigenvalue solver to find the eigenvalue near the expected natural frequency. Use Equation 5-3 to calculate an initial guess for the angular natural frequency and then guide the solver to search eigenvalues near $i\omega_0$. Alternatively, you can first solve the undamped eigenfrequency f_0 (select **Eigenfrequency analysis** and **Eigenfrequency solver** in the **Solver Parameters** dialog box), and then search the damped eigenmodes near $i2\pi f_0$. This approach is also valid for more complex geometries where you do not have an analytical expression for the estimated natural frequency.

This model, which consists of second-order (structural) and first-order (thermal) eigenvalue problems, is numerically quite challenging. You can improve the accuracy of the result by manually scaling the dependent variables. This is an approach that you can use more generally, also: Once you know the magnitudes of the dependent variables, go to the **Advanced** page in the **Solver Parameters** dialog box and define manual scaling of the dependent variables.

An implication from the use of the eigenfrequency solver is that the solution represents variations that take place at the frequency denoted by the eigenvalue. The solution does not contain any stationary information about the system. Therefore, it is natural to interpret the temperature in Equation 5-9 as the thermoelastic variation: It is initialized to zero and the thermal strain is calculated with reference to zero. However, T_{init} in Equation 5-10 and Equation 5-13 is always the true temperature of the beam at rest.

For the 3D problem, you can utilize the symmetry. The beam is truncated to 10 um in the width direction. Thus you get a smaller mesh size, but it also makes the problem easier to solve, because the symmetry condition also prevents any flexural vibrations in this direction.

This example uses both rectangular and brick meshes. In 2D, first draw the geometry and use a mapped mesh to mesh it. In 3D, you utilize work planes by drawing a 2D view of the beam into the work plane. Then mesh it using the mapped mesh, and finally, a mesh extrusion creates the true 3D geometry.

References

1. A. Duwel, R.N. Candler, T.W. Kenny, M. Varghese, "Engineering MEMS Resonators with Low Thermoelastic Damping," Submitted to *JMEMS*, April 2005.

2. S. Gupta, *Estimation of Thermo-Elastic Dissipation in MEMS*, MSc. Thesis, Department of Mechanical Engineering, Indian Institute of Science, Bangalore - 560 012, India. July 2004.

3. T.V. Roszhart, "The Effect of Thermoelastic Internal Friction on the Q of Micromachined Silicon Resonators," *Tech. Dig. Solid-State Sens. Actuator Workshop*, Hilton Head, SC, 1990, pp.13-16.

4. R. Lifshitz., K.L. Roukes, "Thermoelastic Damping in Micro- and Nanomechanical Systems," *Physical Review B*, vol 61, no. 8, 15. Feb. 2000-II

Modeling Using the Graphical User Interface—3D Example

MODEL NAVIGATOR

1 In the Model Navigator select **3D** in the **Space dimension** list.

2 Click **Multiphysics**.

3 From the **Application modes** tree, select **MEMS Module>Solid, Stress-Strain>Transient analysis**.

4 Click **Add**.

5 From the **Application modes** tree, select **COMSOL Multiphysics>Heat Transfer>Conduction>Transient analysis**.

6 Click **Add**.

7 Click **OK**.

OPTIONS AND SETTINGS

1 Select **Options>Constants** and type constants according to the following table:

TABLE 5-3: CONSTANTS

NAME	EXPRESSION
E	157e9
rho	2330
nu	0.3
alpha	2.6e-6
Cp	700
k	90
Tinit	300
w0	4.730^2*sqrt(E/(rho*12))*12e-6/400e-6^2

Constant `w0` is the analytical estimate for the natural angular frequency. It is used as an initial guess for the eigenfrequency solver.

GEOMETRY MODELING

1. Select **Draw>Work Plane Settings**.
2. In the **Quick** page select the **z-x** check box.
3. Click **OK**. The 2D work plane becomes active.
4. Shift-Click the **Rectangle/Square** toolbar button.
5. In the **Width** field type `12e-6`, and in the **Height** field type `400e-6`.
6. Click **OK**, then click **Zoom Extents**.

MESH GENERATION

1. In the work plane select **Mesh>Map Mesh**.
2. In the **Subdomain** page select the first subdomain and click the **Boundary** tab.
3. Select boundary **1**.
4. Select **Constrained edge element distribution** and type 4 for **Number of edge elements**.
5. Select boundary **2**.
6. Select **Constrained edge element distribution** and type 25 for **Number of edge elements**.
7. Click **Remesh**, then click **OK**.
8. Select **Mesh>Extrude Mesh**.
9. In the **Distance** field type `10e-6`.
10. Change to the **Mesh** page
11. Type 2 for **Number of element layers**.
12. Click **OK**.

The 3D geometry should open automatically.

PHYSICS SETTINGS

Subdomain Settings

1. From the **Multiphysics** menu select **1 Geom1: Solid, Stress-Strain (smsld)**.
2. Select **Physics>Subdomain Settings**.

3 Select subdomain **1**, and in the **Material** page give settings according to the following table:

TABLE 5-4: MATERIAL PROPERTIES

PROPERTY	VALUE
E	E
ν	nu
α	alpha
ρ	rho
α_{dM}	0
β_{dK}	0

4 Change to the **Load** page.

5 Select **Include thermal expansion** and type T for **Temp** and 0 for **Tempref**.

6 Click **OK**.

7 From the **Multiphysics** menu select **2 Geom1: Heat Transfer by Conduction (ht)**.

8 Select **Physics>Subdomain Settings**.

9 See that subdomain **1** is selected and in the **Material** page give settings according to the following table:

TABLE 5-5: MATERIAL PROPERTIES

PROPERTY	VALUE
δ_{ts}	1
k (isotropic)	k
ρ	rho
C_p	Cp
Q	-E*alpha*Tinit/(1-2*nu)*(uxt+vyt+wzt)

10 Click **OK**.

Boundary Conditions

1 From the **Multiphysics** menu select **1 Geom1: Solid, Stress-Strain (smsld)**.

2 Select **Physics>Boundary Settings**.

3 Select boundaries **1** and **6**.

4 In the **Constraint** page, see that **Standard notation** is selected and check **Rx**, **Ry**, and **Rz** check boxes.

5 Select boundary **2**.

6 Check the **Ry** check box. This is the symmetry condition.

7 Click **OK**.

8 From the **Multiphysics** menu select **2 Geom1: Heat Transfer by Conduction (ht)**.

9 Select **Physics>Boundary Settings**.

10 See that **Thermal insulation** is the default value.

11 Click **OK**.

COMPUTING THE SOLUTION

1 Select **Solve>Solver Parameters**.

2 Select **Eigenvalue** from the **Solver** list.

3 See the **General** page.

4 Type `1` for **Desired number of eigenvalues**.

5 Type `i*w0` for **Search for eigenvalues around**.

6 From the **Matrix symmetry** list select **Nonsymmetric**.

7 Go to the **Advanced** page.

8 Locate the **Scaling of variables** area and choose **Manual** for **Type of scaling**.

9 Type: `u 1e-4 v 1e-4 w 1e-4 T 1` in the **Manual scaling** edit field.

10 Click **OK**.

11 Click **Solve** in the Main toolbar.

The solver finds a solution with an eigenvalue lambda of roughly 217 + i*3.98e6. The imaginary part is close to the estimated ω_0 value: 3.976e6.

POSTPROCESSING AND VISUALIZATION

To create the Figure 5-2 follow the steps below:

1 Click the **Plot Parameters** toolbar button.

2 On the **General Page** see that **Boundary**, **Deformed shape**, and **Geometry edges** are selected, and all other are cleared.

3 Change to the **Boundary** page and select **Temperature (ht)** from the **Predefined quantities** list.

4 Click **OK**.

You can see the eigenvalue in the postprocessing plot. Use the **Global Data Display** dialog box to calculate the natural frequency and the Q-value:

1 Select **Postprocessing>Data Display>Global**.

2 See the selected eigenvalue.

3 Type `imag(lambda)/2/pi` into the **Expression to evaluate** field.

4 Click **Apply**. You find the natural frequency in the report field at the bottom of the main window.

5 Type `imag(lambda)/2/real(lambda)` into **Expression to evaluate** field.

6 Click **Apply**. You find the Q-value in the report field.

You can also use COMSOL Script:

1 Select **File>Export>Export FEM structure as 'FEM'**. This opens COMSOL Script if it is not open yet.

2 At the command line, type `who` to see that the variable `fem` is in the main workspace; type `fem` to see its fields.

3 Extract the eigenvalue by typing `lambda = fem.sol.lambda`, then press Enter.

4 Type `Q = imag(lambda)/2/real(lambda)`, then press Enter.

5 Type `F0 = imag(lambda)/2/pi`, then press Enter.

Modeling Using the Graphical User Interface—2D Example

Start the model in the Model Navigator.

1 In the Model Navigator select **2D**.

2 Click **Multiphysics**.

3 From the **Application modes** tree select **MEMS Module>Plane Stress>Transient analysis**.

4 Click **Add**.

5 From the **Application modes** tree select **COMSOL Multiphysics>Heat Transfer> Conduction>Transient analysis**.

6 Click **Add**.

7 Click **OK**.

OPTIONS AND SETTINGS

1 Select **Options>Constants** and type constants according to the following table:

TABLE 5-6: CONSTANTS

NAME	EXPRESSION
E	157e9
rho	2330

TABLE 5-6: CONSTANTS

NAME	EXPRESSION
nu	0.3
alpha	2.6e-6
Cp	700
k	90
Tinit	300
w0	4.730^2*sqrt(E/(rho*12))*12e-6/400e-6^2

The constant `w0` is the analytical estimate for the natural angular frequency. It serves as an initial guess for the eigenfrequency solver.

GEOMETRY MODELING

1. Shift-click the **Rectangle/Square** toolbar button.
2. In the **Width** edit field type `400e-6`, and in the **Height** edit field type `12e-6`.
3. Click **OK**, then click **Zoom Extents**.

MESH GENERATION

1. In the work plane select **Mesh>Map Mesh**.
2. In the **Subdomain** page select the first subdomain and click the **Boundary** tab.
3. Select boundary **1**.
4. Select **Constrained edge element distribution** and type 5 for **Number of edge elements**.
5. Select boundary **2**.
6. Select **Constrained edge element distribution** and type 25 for **Number of edge elements**.
7. Click **Remesh**, then click **OK**.

PHYSICS SETTINGS

Subdomain Settings

1. From the **Multiphysics** menu select **1 Plane Stress (smps)**.
2. Select **Physics>Subdomain Settings**.

3 Select the subdomain **1**, and in the **Material** page give settings according to the following table:

TABLE 5-7: MATERIAL PROPERTIES

PROPERTY	VALUE
E	E
ν	nu
α	alpha
ρ	rho
thickness	20e-6
α_{dM}	0
β_{dK}	0

4 Change to the **Load** page.

5 Select **Include thermal expansion** and type T for **Temp** and 0 for **Tempref**.

6 Click **OK**.

7 From the **Multiphysics** menu select **2 Heat Transfer by Conduction (ht)**.

8 Select **Physics>Subdomain Settings**.

9 See that the subdomain 1 is selected, and on the **Material** tab enter settings according to the following table:

TABLE 5-8: MATERIAL PROPERTIES

PROPERTY	VALUE
δ_{ts}	1
k (isotropic)	k
ρ	rho
C_p	Cp
Q	-E*alpha*Tinit/(1-2*nu)/(1-nu)*((1-2*nu)*(uxt+vyt)+(1+nu)*alpha*Tt)

10 Click **OK**.

Boundary Conditions

1 From the **Multiphysics** menu select **1 Plane Stress (smps)**.

2 Select **Physics>Boundary Settings**.

3 Select boundaries 1 and 4.

4 In the **Constraint** page, see that **Standard notation** is selected and select **Rx**, and **Ry** checkboxes.

5 Click **OK**.

6 From the **Multiphysics** menu select **2 Heat Transfer by Conduction (ht)**.

7 Select **Physics>Boundary Settings**.

8 See that **Thermal insulation** is the default value.

9 Click **OK**.

COMPUTING THE SOLUTION

1 Select **Solve>Solver Parameters**.

2 Select **Eigenvalue** from the **Solver** list

3 See the **General** page.

4 Type `1` for **Desired number of eigenvalues**.

5 Type `i*w0` for **Search for eigenvalues around**.

6 Go to the **Advanced** page.

7 Locate the **Scaling of variables** area and choose **Manual** for **Type of scaling**.

8 Type: `u 1e-4 v 1e-4 T 1`, in the **Manual scaling** edit field.

9 Click **OK**.

10 Click **Solve** in the Main toolbar.

The solver finds a solution with an eigenvalue lambda of roughly 194 + i*3.96e6. The imaginary part is close to the estimated ω_0 value: 3.976e6.

POSTPROCESSING AND VISUALIZATION

To create the Figure 5-3 follow the steps below:

1 Click the **Plot Parameters** toolbar button.

2 On the **General Page** see that **Boundary**, **Deformed shape**, and **Geometry edges** are selected, and all other are cleared.

3 Change to the **Surface** page and select **Temperature (ht)** from the **Predefined quantities** list.

4 Click **OK**.

You can see the eigenvalue in the postprocessing plot. Use the **Global Data Display** dialog box to calculate the natural frequency and the Q-value:

1 Select **Postprocessing>Data Display>Global**.

2 See the selected eigenvalue.

3 Type `imag(lambda)/2/pi` into **Expression to evaluate** field.

4 Click **Apply**. You find the natural frequency at the bottom of the main window.

5 Type `imag(lambda)/2/real(lambda)` into **Expression to evaluate** field.

6 Click **Apply**. You find the Q-value at the bottom of the main window.

You can also use the COMSOL Script:

1 Select **File>Export>Export FEM structure as 'FEM'**. This opens COMSOL Script if it is not open yet.

2 In the command line, type `who` to see the variable fem is there, and type `fem` to see its fields.

3 Extract the eigenvalue by typing `lambda = fem.sol.lambda`, then press Enter.

4 Type `Q = imag(lambda)/2/real(lambda)`, then press Enter.

5 Type `F0 = imag(lambda)/2/pi`, then press Enter.

Microchannel H-Cell

This example was originally formulated by Albert Witarsa under Professor Bruce Finlayson's supervision at the University of Washington in Seattle. It was part of a graduate course in which the assignment consisted of evaluating the potential of patents in the field of microfluidics through mathematical modeling.

The model treats a so called H-micro cell for separation through diffusion. The cell puts two different laminar streams in contact for a controlled period of time. The contact surface is well-defined and, by controlling the flow rate, it is possible to control the amount of species that are transported from one stream to the other through diffusion.

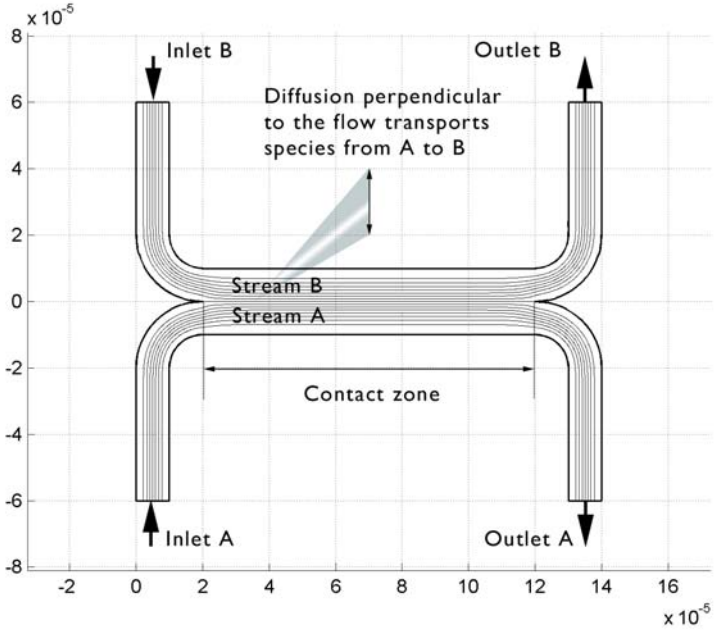


Figure 5-4: Schematic picture of the H-micro cell.

Model Definition

Figure 5-5 shows the geometry of the microcell from Albert Witarsa's and Professor Finlayson's assignment. The cell geometry is divided in half due to symmetry. The

design is done to avoid upsets in the flow field as the two streams, A and B, are united. The reason for this is to avoid the two streams mixing through convection, because this would mix all species equally and you would lose control over the separation abilities. The transport of species from streams A to B should only take place by diffusion, which implies that species with low diffusion coefficients stay in their respective stream.

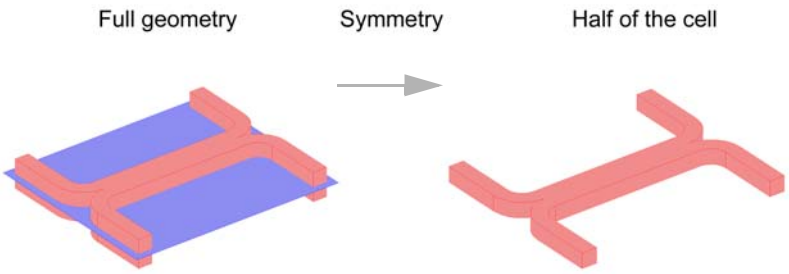


Figure 5-5: Model geometry. Due to symmetry, you only need to model half the cell. The design is also required to smoothly let both streams come in contact with each other. This avoids any type of convective mixing.

The first part of the problem involves the solution of the fluid flow in the H-cell. According to the specifications, the flow rate at the inlet is about 0.1 mm s^{-1}. This implies that the Reynolds number is low and well inside the laminar flow region:

$$\left(Re = \frac{d\rho u}{\mu}\right) \Leftrightarrow Re = \frac{1 \cdot 10^{-5} 1 \cdot 10^{3} 1 \cdot 10^{-4}}{1 \cdot 10^{-3}} \qquad (5\text{-}14)$$

Equation 5-14 gives a Reynolds number of 0.001 for a water solution and the channel dimensions given in Figure 5-4 and is typical for microchannels. This means that it is easy to get a numerical solution of the full momentum balance and continuity equations for incompressible flow with a reasonable number of elements. The equations that you have to solve are the Navier-Stokes equations at steady-state:

$$\begin{gathered} -\nabla \cdot \eta(\nabla \mathbf{u} + (\nabla \mathbf{u})^{T}) + \rho(\mathbf{u} \cdot \nabla)\mathbf{u} + \nabla p = 0 \\ \nabla \cdot \mathbf{u} = 0 \end{gathered} \qquad (5\text{-}15)$$

In these equations, ρ denotes density (kg m^{-3}), $\mathbf{u}$ the velocity vector (m s^{-1}), η denotes viscosity (kg m^{-1} s^{-1}), and p pressure (Pa).

Separation in the H-cell involves species in a relatively low concentration compared to the solvent, in this case water. This means that the solute molecules only interact with water molecules and it is safe to use Fick's law to describe the diffusive transport in the cell. The mass balance equation for a solute is given by the Convection-Diffusion application mode, which solves the following equation

$$-\nabla \cdot (-D\nabla c + c\mathbf{u}) = 0 \tag{5-16}$$

In this equation, D denotes the diffusion coefficient ($m^2\ s^{-1}$) and c concentration (mole m^{-3}). D will vary with a magnitude to simulate the mixing of different species.

In the first approach, assume that a change in concentration of the solute does not influence the density and viscosity of the fluid. This implies that it is possible to solve the Navier-Stokes equations first and then solve the mass balance equation. To solve these equations, you need some boundary conditions. In a second step, the model includes a quadratic dependence of the viscosity on concentration:

$$\eta = \eta_0(1 + c^2) \tag{5-17}$$

The influence of concentration on the viscosity is usually observed in solutions of larger molecules.

Start by looking at the boundary conditions for the Navier-Stokes equations. Figure 5-6 qualitatively explains these boundary conditions.

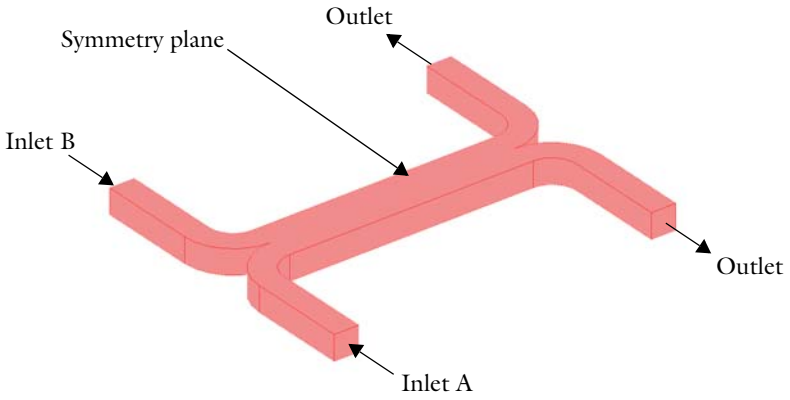


Figure 5-6: Boundary conditions for Navier-Stokes equations.

The normal flow/pressure conditions and the inlets and outlets set the velocity at the boundary perpendicular to the boundary. This is expressed by:

$$\begin{aligned} \mathbf{u} \cdot \mathbf{t} &= 0 \\ p &= p_i \end{aligned} \tag{5-18}$$

where p_i is the pressure. At the outlet the pressure is set to zero, and at the inlet the pressure is given by the pressure drop over the cell. The inlet and outlet conditions comply with the H-cell being a part of a channel system of constant width, which allows for the assumption of developed flow, and that velocity components tangential to the outlet boundary are zero. The symmetry boundary condition means that the velocity component in the normal direction of the surface is zero:

$$\mathbf{u} \cdot \mathbf{n} = 0 \tag{5-19}$$

The no-slip conditions state that the velocity is zero in the x-, y-and z-directions at the wall:

$$(u, v, w) = (0, 0, 0) \qquad \text{at the walls} \tag{5-20}$$

The boundary conditions for the mass balance define the composition of the solution at the inlet:

$$\begin{aligned} c &= c_A \qquad \text{at inlet A} \\ c &= c_B \qquad \text{at inlet B} \end{aligned} \tag{5-21}$$

In the model c_A = 1and c_B = 0. The first value was chosen so that c in Equation 5-17 has correct magnitude.

The walls of the cell are modeled as insulating boundaries, which yields:

$$(-D\nabla c + c\mathbf{u}) \cdot \mathbf{n} = 0 \qquad \text{at the walls} \tag{5-22}$$

This equation states that the flux perpendicular to the boundary is equal to zero. The outlet condition uses the assumption that the convective transport is much larger than the diffusive transport perpendicular to the outlet. This is given in Equation 5-23:

$$(-D\nabla c + c\mathbf{u}) \cdot \mathbf{n} = c\mathbf{u} \cdot \mathbf{n} \qquad \text{at the outlets} \tag{5-23}$$

This equation eliminates concentration gradients in the direction of the flow at the position of the outlet.

Results

Figure 5-7 below shows the velocity field, defined as the modulus of the velocity vector. The flow is symmetric and is not influenced by the concentration field.

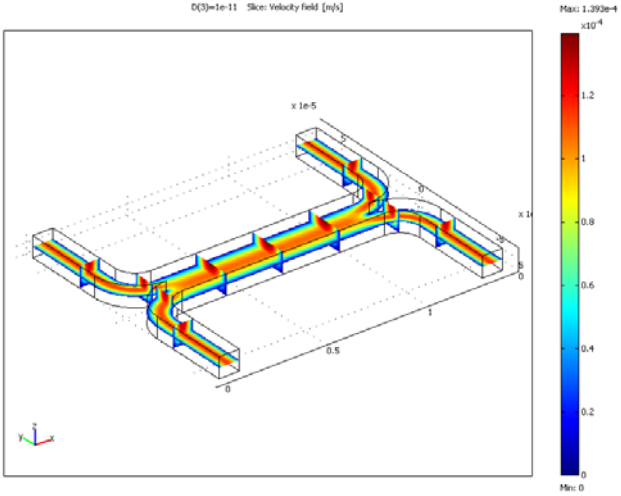


Figure 5-7: Modulus of the velocity vector.

Figure 5-8 shows the concentration distribution for a species with a relatively large diffusivity, the light species in our study.

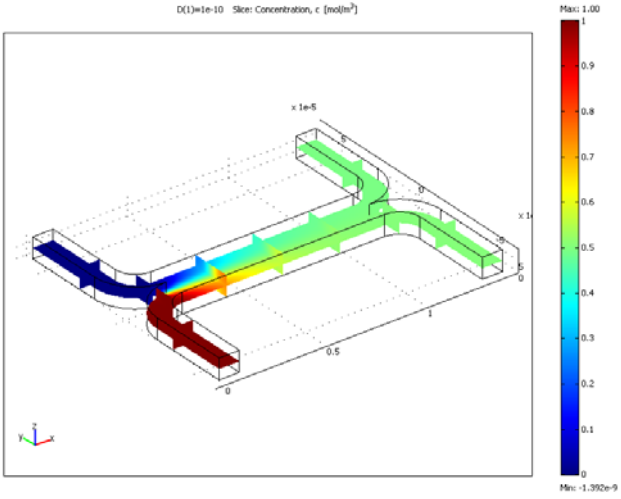


Figure 5-8: Concentration distribution for species with the larger diffusivity.

The calculation shows that the degree of mixing is almost perfect for the lighter species. The species with a diffusion coefficient that is ten times less than the lighter species shows a completely different result.

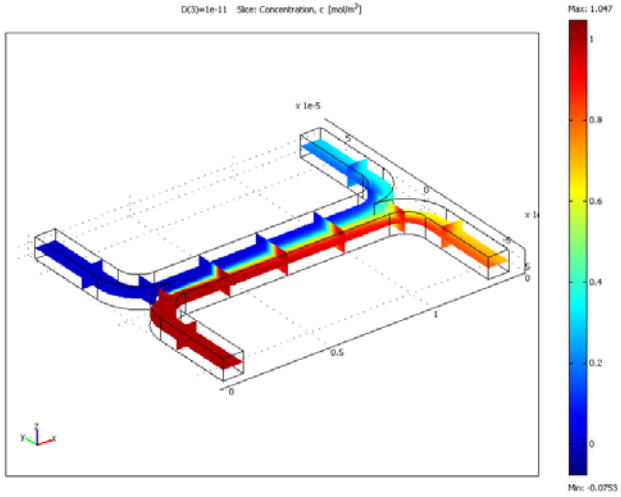


Figure 5-9: Concentration distribution for species with the smaller diffusivity.

The concentration distribution shown in Figure 5-9 shows that the diffusion coefficient for the species is low enough to avoid any significant mixing between streams A and B. The simulation clearly shows that the H-cell is able to separate lighter molecules from heavier ones. A cascade of H-cells can achieve a very high degree of separation.

In some cases, especially those involving solutions of macro-molecules, the concentration of the macro-molecule has a large influence on the viscosity of the liquid. In such a case, it is necessary to solve the Navier-Stokes and convection-diffusion equations simultaneously. Figure 5-10 shows the results of this simulation. You can see that the flow field is largely influenced by changes in the concentration. It is clear from this plot that the velocity becomes asymmetric due to changes in viscosity. As a consequence of the changed flow field, the transport of molecules to the outlet B is also different from the result using a constant flow field. You can see the difference by comparing Figure 5-9 and Figure 5-11.

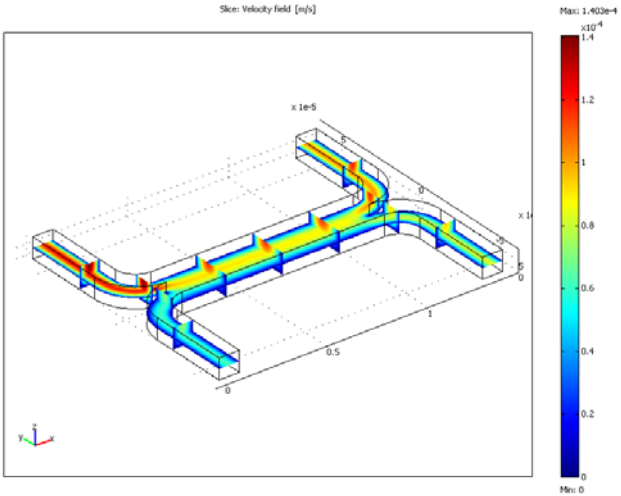


Figure 5-10: Velocity field. The viscosity varies with the concentration according to $\eta = \eta_0(1+c^2)$. *The figure shows that the velocity field is greatly affected by the variations in concentration. Compare to the velocity field in Figure 5-7.*

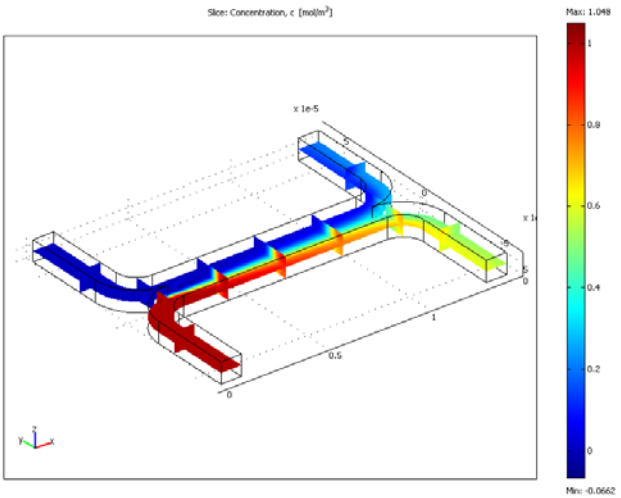


Figure 5-11: Concentration distribution for species with the smaller diffusivity, when the viscosity of the fluid varies with the concentration according to $\eta = \eta_0(1+c^2)$. *This figure shows that less species are transported to the outlet B than what is seen in Figure 5-9.*

Modeling in COMSOL Multiphysics

This example uses the predefined multiphysics application mode found in the **Model Navigator** branch **MEMS Module>Microfluidics>Flow with Species Transport> Incompressible Navier-Stokes**. This selection adds the Incompressible Navier-Stokes and the Convection and Diffusion application modes to the model.

The origin of the Convection and Diffusion application mode depends on whether your license includes the Chemical Engineering Module. This example was created without the module, and thus the step-by-step instructions use **Convection and Diffusion (cd)** application mode. However, if you have the Chemical Engineering Module, you will use **Convection and Diffusion (chcd)** application mode instead.

Originally, this predefined multiphysics application mode has a one-way coupling: The velocity components of the Incompressible Navier-Stokes are prewritten in the corresponding edit fields of the Convection and Diffusion application mode. In the second study of the model you create a two-way coupling and write a concentration-dependent expression for the viscosity of the fluid. Thus, in the first study you can solve the application modes sequentially, but in the second study you need to solve both application modes simultaneously.

Modeling Using the Graphical User Interface

1. Start COMSOL Multiphysics.
2. In the **New** page, select **3D** in the **Space dimension** list.
3. Select **MEMS Module>Microfluidics>Flow with Species Transport>Incompressible Navier-Stokes**.
4. Click **OK**.

OPTIONS AND SETTINGS

1. Select **Constants** from the **Options** menu.
2. Enter the following constants in the **Constants** dialog box:

NAME	EXPRESSION
eta	1e-3
rho	1e3
p0	2
D	1e-11
c0	1

3 Click **OK**.

GEOMETRY MODELING

1 From the **Draw** menu, select **Work Plane Settings**.

2 Click **OK** to create the default work plane (the x-y plane at z=0).

3 Open the **Axes/Grid Settings** from the **Options** menu.

4 On the **Axis** page, enter axis settings according to table below.

PROPERTY	VALUE
x min	-1
x max	15
y min	-7
y max	7

5 Click the **Grid** tab, clear the **Auto** check box, and set both **x spacing** and **y spacing** to 1.

6 Click **OK**.

7 Click the **Line** button in the Draw toolbar.

8 Click on the points with the corresponding x- and y-coordinates `(0,6)`, `(0,2)`.

9 Select the **2nd Degree Bezier Curve** tool and click on the coordinates `(0,0)` and `(2,0)`.

10 Select the **Line** tool and click on the coordinates `(12,0)`.

11 Select the **2nd Degree Bezier Curve** tool and click on the coordinates `(14,0)` and `(14,2)`.

12 Select the **Line** tool and click on the coordinates `(14,6)`.

13 Continue with the **Line** tool and click on the coordinates `(13,6)`, `(13,2)`.

14 Select the **2nd Degree Bezier Curve** tool and click on the coordinates `(13,1)` and `(12,1)`.

15 Select the **Line** tool and click on the coordinates `(2,1)`.

16 Select the **2nd Degree Bezier Curve** tool and click on the coordinates `(1,1)` and `(1,2)`.

17 Select the **Line** tool and click on the coordinates `(1,6)`.

18 Click the right mouse button anywhere in the drawing area. This will close the set of lines and create a solid object.

You have now created one half of the 2D cross-section and can use copy, paste, and rotate to create the other half:

1. Click the **Copy** button in the Main toolbar.
2. Click the **Paste** button. Let the **Displacements** remain `0` and click **OK**.
3. Click the **Rotate** toolbar button and enter `180` in the α edit field in the **Rotation angle** area.
4. Set the point `(7,0)` as **Center point**.
5. Click **OK**.
6. Choose **Select All** in the **Edit** menu.
7. Click the **Create Composite Object** button.
8. Clear the **Keep interior boundaries** check box.
9. Click **OK**.

The cross section is now finished, and you can extrude it to the full 3D geometry:

10. Select **Extrude** from the **Draw** menu. Let **Distance** remain `1`.
11. Click **OK**.

The last step is to scale the geometry to the correct scale.

1. Click the **Scale** button.
2. Enter `1e-5` in all **Scale factor** edit fields.
3. Click **OK**.
4. Click **Zoom Extents**.

PHYSICS SETTINGS

Subdomain Settings

1. Select **Geom1: Incompressible Navier-Stokes (mmglf)** from the **Multiphysics** menu.
2. Select **Subdomain Settings** from the **Physics** menu.
3. Select subdomain 1 and specify the fluid properties according to the following table:

SUBDOMAIN	1
ρ	`rho`
η	`eta`
Fx	0
Fy	0

4 Click **OK**.

5 Select **Geom1: Convection and Diffusion (cd)** from the **Multiphysics** menu.

6 Select **Subdomain Settings** from the **Physics** menu.

7 Specify the subdomain settings given in the following table. Note that you only need to define **D isotropic**. The other settings are correct by default.

SUBDOMAIN	1
D isotropic	D
R	0
u	u
v	v
w	w

8 Click **OK**.

Boundary Conditions

1 Select **Geom1: Incompressible Navier Stokes (mmglf)** from the **Multiphysics** menu.

2 Select **Boundary Settings** from the **Physics** menu.

3 Specify boundary conditions according to the table below:

BOUNDARY	2, 8	20, 22	4	ALL OTHER
Type	Normal flow/ Pressure	Normal flow/ Pressure	Slip/ Symmetry	No-slip
p	p0	0	-	-

4 Click **OK**.

5 Select **Geom1: Convection and Diffusion (cd)** from the **Multiphysics** menu.

6 Select **Boundary Settings** from the **Physics** menu.

7 Specify boundary conditions according to the table below:

BOUNDARY	2	8	20, 22	ALL OTHER
Type	Concentration	Concentration	Convective flux	Insulation/ symmetry
c_0	c0	0	-	-

8 Click **OK**.

MESH GENERATION

1 Select **Mesh Parameters** from the **Mesh** menu.

2 Select **Finer** from the **Predefined mesh sizes** list.

3 Click the **Remesh** button and then **OK**.

COMPUTING THE SOLUTION

We can start by first computing the solution for the velocity field and then use that solution for solving the mass transport problem.

1 Click the **Solver Manager** button in the Main toolbar.

2 On the **Initial value** tab, click the **Initial value expression** button.

3 Click the **Solve For** tab, select **Incompressible Navier-Stokes (mmglf)** only.

4 Click **OK**.

The default solver for Incompressible Navier-Stokes application mode is Stationary nonlinear solver, which is the correct choice at this stage. You can verify this setting in the following steps 5–7, or you can jump to step 8 to solve the model.

5 Open the **Solver Parameters** dialog box from the **Solve** menu.

6 See that **Stationary nonlinear** is selected in the **Solver** list.

7 Click **OK**.

8 Click the **Solve** button in the Main toolbar.

You have now computed the velocity field and can use that solution to make a parametric analysis for the mass transport problem with varying diffusivity:

1 Click the **Solver Manager** button.

2 On the **Solve For** tab, select **Convection and Diffusion (cd)** only.

3 Click the **Initial Value** tab and then click the **Current solution** button in the **Initial value** area.

4 Click **OK**.

5 Open the **Solver Parameters** window and select **Parametric nonlinear** from the **Solver** list.

6 Enter `D` in the **Name of parameter** edit field.

7 Edit `1e-10 5e-11 1e-11` in the **List of parameter values** edit field.

8 Click **OK**.

9 Click the **Solve** button in the Main toolbar.

POSTPROCESSING AND VISUALIZATION

Instructions to create Figure 5-7:

1. Open the **Plot Parameters** dialog box from the **Postprocessing** menu.
2. Go to the **General** page.
3. Check that only the **Slice** and **Geometry Edges** check boxes are selected.
4. Click the **Slice** tab.
5. Set **Slice data** to **Velocity field (mmglf)** from the **Predefined quantities** list.
6. Under **Slice positioning**, set **x levels** to 5, **y levels** to 2 and **z levels** to 1.
7. Click **OK**.

Instructions to create Figure 5-8:

1. Open the **Plot Parameters** dialog box.
2. Go to the **General** page, select **1e-10** from the **Parameter value** list.
3. Go to the **Slice** page and select **Concentration, c (cd)** from the **Predefined quantities** list.
4. Click **OK**.

Figure 5-9 is made in the same way as Figure 5-8 but for the parameter value 1e-11.

PHYSICS SETTINGS—STUDY 2

1. Select **1 Incompressible Navier-Stokes (mmglf)** from the **Multiphysics** menu.
2. Open the **Subdomain Settings** window.
3. Select the subdomain 1 and change the viscosity expression from `eta` to `eta*(1+c^2)`.
4. Click **OK**.

COMPUTING THE SOLUTION—STUDY 2

Now, when the viscosity depends on the concentration, you must solve for all equations simultaneously:

1. Click the **Solver Manager** button in the Main toolbar.
2. On the **Initial value** tab, click the **Initial value expression** radio button.
3. Go to the **Solve** page and select **Geom1 (3D)** from the **Solve for variables** list.
4. Click **OK**.
5. Open the **Solver Parameters** dialog box from the **Solve** menu.

6 Select **Stationary nonlinear** from the **Solver** list.

7 Click **OK**.

8 Click the **Solve** button in the Main toolbar.

POSTPROCESSING AND VISUALIZATION—STUDY 2

To create Figure 5-10, do exactly as for Figure 5-7.

To create Figure 5-11, follow the steps for Figure 5-8. However, you do not need to set the parameter values, because the parametric solver was not used for this solution.

Electroosmotic Flow in a Biochip

Miniature laboratories[1] are required to efficiently analyze the information in human DNA, and such devices could facilitate tailor-made diagnosis and treatment of hereditary diseases for each individual. One problem that arises in these lab-on-a-chip devices lies in the transport of the liquid samples and other solutions in the biochip, which are of very small dimensions. Moving parts in this micrometer scale make the biochips very expensive and fragile and should therefore be avoided if possible. One way of transporting the fluid in the samples is through the use of electrokinetic effects, where the charged ions in the solutions are subjected to an electric field. These ions are able to drag the entire solution through the channels in the microchip from one analysis point to the others.

Two mechanisms can drive the flow of a saline solution in an electric field. In the presence of solid surfaces, like the micromachined surfaces of the channels of a biochip, a charged solution is formed close to the wall surfaces. This layer is usually denoted the *diffuse double layer* and is formed due to the negatively or positively charged groups on the wall's surfaces, depending on the material used. The electric field displaces the charged liquid in the charged double layer generating a so-called electroosmotic flow. Figure 5-12 shows the velocity field shortly after the application of the electric field.

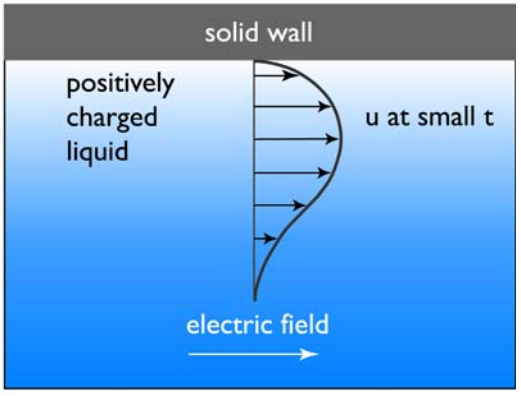


Figure 5-12: Velocity field near a solid wall. The fluid flows in the direction of the electric field.

1. This model was submitted by Dr. Jordan MacInnes, University of Sheffield.

A force is imposed on the positively-charged solution close to the wall surface, and the fluid starts to flow in the direction of the electric field. The gradients in velocity perpendicular to the wall give rise to viscous transport in this direction. In the absence of other forces, the velocity profile eventually becomes almost uniform in the cross-section perpendicular to the wall. The time development of the overall flow is around 1 ms, while the flow in the double layer responds to changes far more rapidly. In the layer model of MacInnes (Ref. 5), the double layer is replaced with the Helmholtz-Smoluchowski relationship between wall velocity and wall electric field (Ref. 7).

The second effect arises due to differences in mobility and charge of the ionic species. Negatively- and positively-charged ions migrate in different directions, and these ions drag water molecules to different extents through the channel system. Water is dragged by the sodium ions, which can coordinate more water molecules than the chloride ions in the solution. This is denoted electrophoretic flow, and the principle is depicted in Figure 5-13.

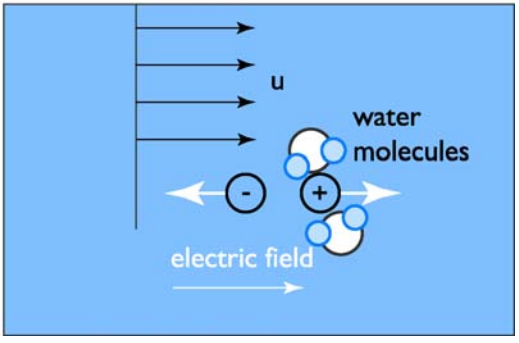


Figure 5-13: Electrophoretic flow.

The model in this section was developed by Dr. Jordan MacInnes at the University of Sheffield in the UK. The modeling work at his department has been combined with experimental studies.

This particular model does not include the mass balances of dissolved species in the sample (see Ref. 5), but you could easily add these through the Nernst-Planck application mode (in the Chemical Engineering Module) or the Convection-Diffusion application mode. Furthermore, using a time-dependent expression for the voltage at the inlet and outlet boundaries makes the problem time-dependent.

Model Definition

The flow in the chip is given by the electroosmotic effect just described. In order to simulate this type of flow, the model must couple the electric potential distribution in the ionic solution in the chip and the equations for fluid flow.

Figure 5-14 shows the chip's geometry. The imposed potential difference between its different parts produces a flow in the vertical or horizontal direction depending on the imposed field. Mode A generates a horizontal flow from right to left in the main channel of the chip; Mode B generates a flow that is vertical in the vertical channels and horizontal, from right to left, in the part of the main channel that unites the vertical branches.

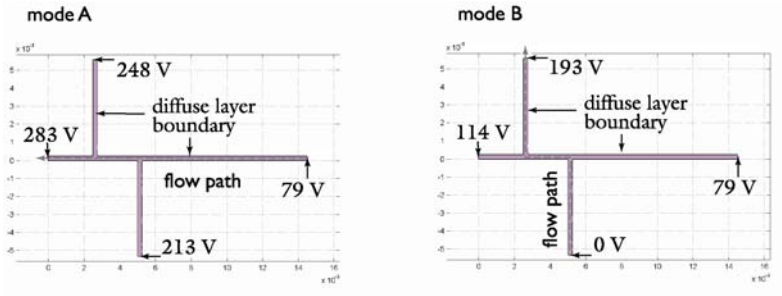


Figure 5-14: Geometry and electric field setup in the two different modes: in Mode A the flow is expected in the horizontal branch, and mode B has flow in the vertical branch.

The voltage is set at the open boundaries where the fluid is allowed to enter or leave the channel system. The wall boundaries are denoted diffusive layer boundaries. From the published work of Dr. MacInnes (Ref. 5), you can expect the flow to be laminar and of low Reynolds number. This implies that you can use the Stokes Flow application mode. The Stokes flow equations are almost the same as the Navier-Stokes equations with the exception that they assume that the inertial term $\rho(\mathbf{u} \cdot \nabla)\mathbf{u}$ is zero. Therefore they describe flow whose Reynolds number is very low and where the inertial forces in the fluid are very small. The system of equations is almost linear in the Stokes case, as opposed to Navier-Stokes.

The Stokes flow equations in the *total stress tensor* formulation are

$$\rho\frac{\partial \mathbf{u}}{\partial t} - \nabla \cdot [-p\mathbf{I} + \eta(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)] = \mathbf{F}$$
$$\nabla \cdot \mathbf{u} = 0 \qquad (24)$$

where η denotes the dynamic viscosity (kg/m · s), $\mathbf{u}$ is the velocity vector (m/ s), ρ denotes the fluid's density (kg/m^3), and p is the pressure (Pa).

The boundary conditions, according to the notations just given, are the following:

$$\mathbf{u} = \frac{\varepsilon_w \varepsilon_0 \zeta_0}{\eta} \nabla V \qquad \text{diffuse layer, wall}$$

$$[-p\mathbf{I} + \eta(\nabla \mathbf{u} + (\nabla \mathbf{u})^T)] \cdot \mathbf{n} = 0 \qquad \text{voltage boundary, inlet or outlet}$$

where ε_w denotes the permittivity of water and ε_0 the permittivity of free space (A s $V^{-1}m^{-1}$), ζ_0 denotes the *zeta-potential* at the wall of the channel (V), and V denotes the potential (V). For a more detailed review of the diffuse layer wall boundary condition, see Ref. 5 and Ref. 6.

Assuming that there are no concentration gradients in the ions that carry the current, the current balance in the channel can be expressed through Ohm's law and the balance equation for current density:

$$\nabla \cdot (-\sigma \nabla V) = 0$$

where σ denotes the conductivity (S/m), while the expression within the brackets represents the current density (A/m^2).

The corresponding boundary conditions for the current balance are

$$-\sigma \nabla V \cdot \mathbf{n} = 0 \qquad \text{diffuse layer, wall}$$

$$V = V_0 \qquad \text{voltage boundary, inlet or outlet}$$

where V_0 corresponds to the voltage shown in the previous figure at the neutral boundaries. At these boundaries, the potential and current distribution in the chip determine if the fluid enters or exits the chip or stays at rest.

The boundary condition for the flow at the wall boundaries sets the velocity vector proportional to the gradient of the potential.

The input data used in this model is listed in the following table. The constants represent the dynamic viscosity, density, conductivity, electrical permittivity, and the zeta-potential of the ionic solution, respectively.

PROPERTY	VALUE
η	1e-3 kg m^{-1} s^{-1}
ρ	1e3 kg m^{-3}
σ	0.11845 S m^{-1}
ε_w	80.2 A s V^{-1} m^{-1}
ζ_0	0.1 V

Results

Figure 5-15 shows the potential distribution for Mode A in the chip. From this plot it is clear that the largest potential differences are in the horizontal direction. This also implies that the main flow is in the same direction.

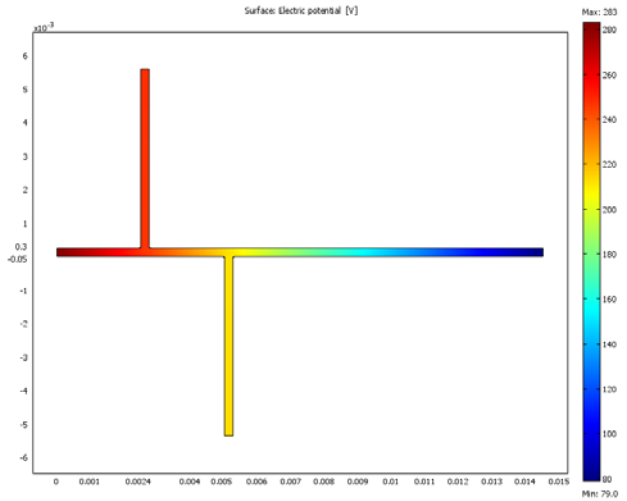


Figure 5-15: Potential distribution in the chip when electric field from mode A is applied.

The plot in Figure 5-16 shows that the average velocity in the channel is approximately 1 mm/s. The largest velocity is at the corner walls where the electric field is large. This clearly shows the effect of the driving force located at the walls. In normal pressure-driven flow, the velocity at the solid surfaces is zero. In fact, looking at the pressure field, you would find that it is constant in the channels.

Studying the velocity flow lines in the channel, Figure 5-17 shows that flow takes a small deviation at the T-junctions in the chip. This can also be detected in the color scale of the velocity, which decreases in the middle of the junction.

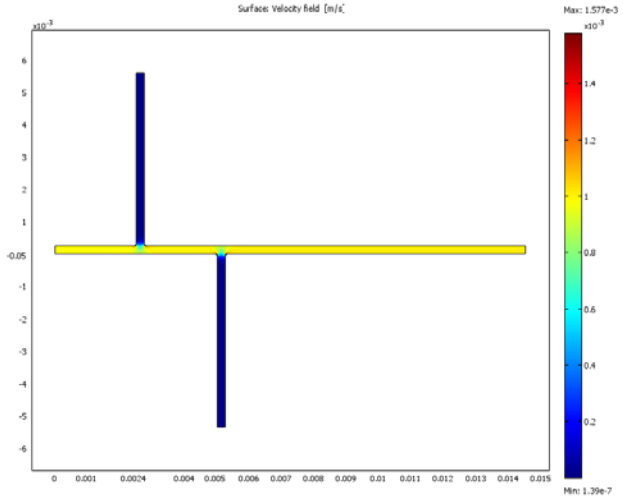


Figure 5-16: Velocity distribution in the chip with the electric field from Mode A applied.

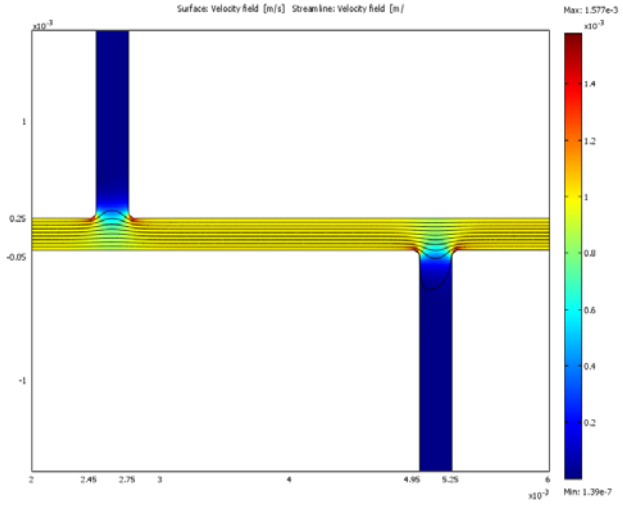


Figure 5-17: Close-up of the velocity field. The surface color is given by the modulus of the velocity vector and the flow lines by the velocity field.

Regulating the potential can quickly and efficiently change the path of the flow in the channels. This makes it possible to mix different solutions in different branches of the chip. Figure 5-18 shows a different flow direction with the same magnitude as the potential configuration given in Mode A. In this plot you can also find the corner effects and the low-velocity regions in the T-junctions, as the channel width increases.

Figure 5-19 shows that the flow decreases in the outer parts of the turns while a maximum is seen in the inner parts. This maximum is actually larger in this case compared to the first case. This is due to the fact that the electric field is much stronger around the corners compared to the straight path in the first simulation.

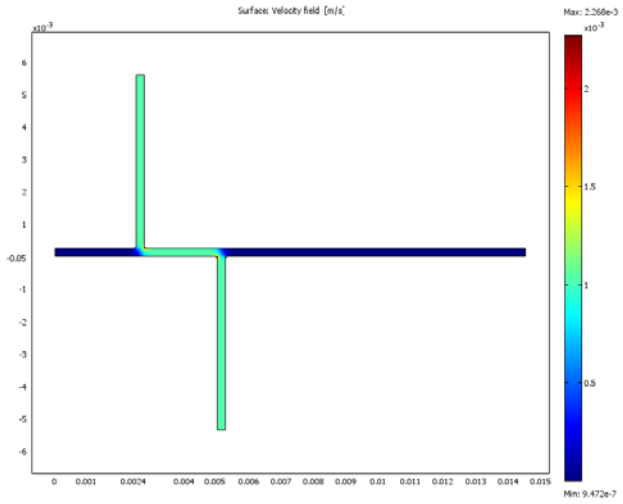


Figure 5-18: Flow distribution in the chip with the electric field from Mode B applied.

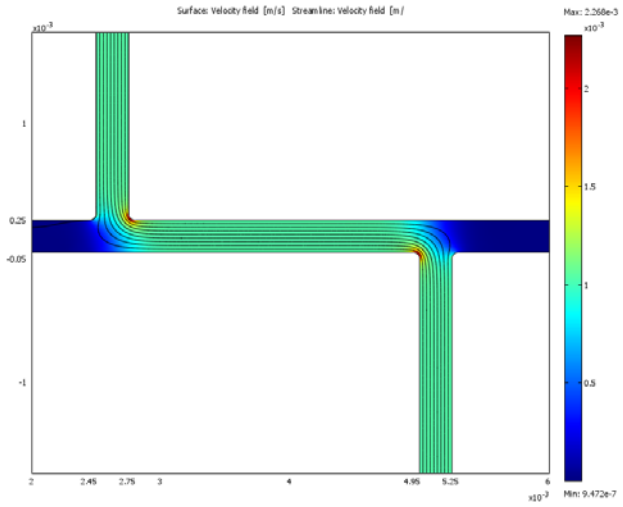


Figure 5-19: Close-up of the flow distribution near the T-junctions.

References

5. J.M. MacInnes, "Computation of Reacting Electrokinetic Flow in Microchannel Geometries," *Journal of Chemical Engineering Science*, 2002.

6. W. Menz, J. Mohr, and O. Paul, *Microsystems Technology*, Wiley-VCH Verlag GmbH, 2001.

7. R.F. Probstein, *Physicochemical Hydrodynamics*, Wiley-Interscience, 1994.

Modeling Using the Graphical User Interface

1. In the **Model Navigator** select **2D** in the **Space dimension** list.
2. Choose the application mode **MEMS Module>Microfluidics>Stokes Flow**.
3. Click the **Multiphysics** button.
4. Click the **Add** button to add the application mode to the model.
5. Select the application mode **MEMS Module>Conductive Media DC** and click **Add**.
6. Click **OK**.

OPTIONS AND SETTINGS

1 Select **Constants** from the **Options** menu.

2 Define the following constants:

NAME	EXPRESSION
eta	1e-3
rho	1e3
k1	0.11845
epsw	80.2
zet0	0.1

3 Click **OK**.

GEOMETRY MODELING

You can import the geometry by selecting **File>Import>CAD Data From File**. Select that file and then click **OK** to import it.

The following steps also show how you can create the geometry using the CAD tools in COMSOL Multiphysics.

1 Press the Shift key and click the **Rectangle/Square** button on the Draw toolbar.

2 In the **Rectangle** dialog box that appears, enter these properties, then click **OK**:

PARAMETER	VALUE
Width	0.25e-3
Height	5.35e-3
x position	2.5e-3
y position	2.5e-4

3 Click the **Zoom Extents** button on the Main toolbar.

4 Repeat this procedure for two more rectangles with the dimensions in the following table:

RECTANGLE	1	2
Width	0.25e-3	1.45e-2
Height	5.35e-3	2.5e-4
x position	5e-3	0
y position	-5.35e-3	0

5 Click the **Zoom Extents** button on the Main toolbar.

6 Select **Axes/Grid Settings** from the **Options** menu.

7 Click the **Grid** tab and clear the **Auto** check box. Then enter `0.001` in both the **x spacing** and **y spacing** edit fields.

8 Enter `2.5e-3 2.45e-3 2.75e-3 2.8e-3 4.95e-3 5.25e-3 5.3e-3` in the **Extra x** edit field, then enter `-5e-5 0.25e-3 3e-4` in the **Extra y** field.

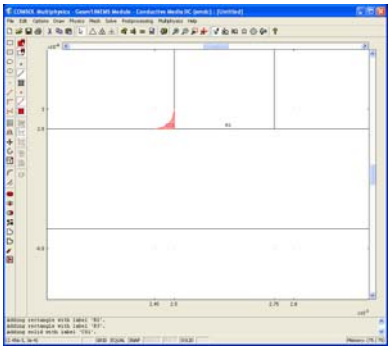
9 Click **OK**.

10 Use the **Zoom Window** tool to zoom in on the inverted T-junction.

11 Select the **2nd Degree Bezier Curve** tool and click on the points `(2.45e-3 2.5e-4)`, `(2.5e-3 2.5e-4)`, and `(2.5e-3 3e-4)`.

12 Click the **Line** button, then click on the corner `(2.5e-3 2.5e-4)`.

13 Click the right mouse button to create a solid composite object.



14 In a similar fashion (click near the grid points near the corner), smooth the right corner of this junction.

15 Click **Zoom Extents** and then use the **Zoom Window** tool to zoom in on the T-junction to the right.

16 Create smoothed corners for this junction in the same way you did for the inverted T-junction corners.

17 Click **Zoom Extents**.

18 Press Ctrl+A to select all geometry objects.

19 Click the **Union** button on the Draw toolbar and finally the **Delete Interior Boundaries** button.

PHYSICS SETTINGS

Boundary Conditions

1 From the **Multiphysics** menu select **1 Stokes Flow (mmglf)**.

2 Open the **Boundary Settings** dialog box from the **Physics** menu.

3 Enter the boundary conditions according to the following table. Click the **Electroosmotic Velocity** tab to specify the electroosmotic settings.

BOUNDARY	1	2–4, 6–8, 10, 11, 13–16	5, 9, 12
Boundary condition	Outflow/ Pressure	Electroosmotic velocity	Neutral
p_0	0		
Ex		Ex_emdc	
Ey		Ey_emdc	
ζ		zet0	

4 Click **OK**.

5 Select **2 Conductive Media DC (emdc)** from the **Multiphysics** menu.

6 Open the **Boundary Settings** dialog box from the **Physics** menu.

7 Enter the boundary conditions according to this table:

BOUNDARY	1	5	9	12	2-4, 6-8, 10, 11, 13-16
Boundary condition	Electric potential	Electric potential	Electric potential	Electric potential	Electric insulation
V_0	283	248	213	79	

8 Click **OK**.

Subdomain Settings

1 From the **Multiphysics** menu select **1 Stokes Flow (mmglf)**.

2 Open the **Subdomain Settings** dialog box from the **Physics** menu.

3 Enter the subdomain settings according to the following table:

SUBDOMAIN	1
ρ	rho
η	eta
F_x	0
F_y	0

4 Click to the **Microfluidic** tab and type `epsw` in the **Relative permittivity** edit field.

5 Click **OK**.

6 Switch to the **Conductive Media DC** application mode and enter these subdomain settings:

SUBDOMAIN	1
σ (isotropic)	k1
Q_j	0
$\mathbf{J}^e$	[0 0]
d	1

7 Click **OK**.

MESH GENERATION

1 Select **Mesh Parameters** from the **Mesh** menu.

2 Type `0.6` in the **Mesh curvature factor** edit field.

3 Click the **Boundary** tab.

4 Select boundaries 1, 5, 9, and 12 from the **Boundary selection** list.

5 Type `1e-4` in **Maximum element size** edit field.

6 Click the **Remesh** button to initialize the mesh and click **OK** to finish.

COMPUTING THE SOLUTION

Click the **Solve** button on the Main toolbar.

POSTPROCESSING AND VISUALIZATION

Follow these instructions to generate Figure 5-15:

1 Open the **Plot Parameters** dialog box from the **Postprocessing** menu.

2 On the **General** tab, make sure that only the **Surface** and **Geometry edges** plot types are selected.

3 On the **Surface** page, select **Electric potential (emdc)** from the **Predefined quantities** list on the **Surface Data** tab.

4 Click **OK**.

To create Figure 5-16 and Figure 5-17, follow these steps:

1 Open the **Plot Parameters** dialog box from the **Postprocessing** menu.

2 On the **General Page**, make sure that only the **Surface** and **Geometry edges** plot types are selected.

3 On the **Surface** page select **Velocity field (mmglf)** from the **Predefined quantities** list on the **Surface Data** tab.

4 Click **Apply;** Figure 5-16 results.

5 On the **General** page select the plot types **Surface**, **Streamline** and **Geometry edges**.

6 Click the **Streamline** tab and select **Velocity field (mmglf)** in the **Predefined quantities** list on the **Streamline Data** tab.

7 On the **Start Points** tab, click the **Specify start point coordinates** button. Enter `linspace(2e-3,2e-3,10)` in the **x** edit field, then enter `linspace(0,0.25e-3,10)` in the **y** edit field.

8 Click the **Line Color** tab and select the **Uniform color** option. Then click the **Color** button and select black. Click **OK**.

9 Click the **Advanced** button.

10 Set the **Maximum number of integration steps** to 4000.

11 Click **OK** to close the **Advanced Streamline Parameters** dialog box.

12 Click **OK** in the **Plot Parameters** dialog box.

13 Select **Axes/Grid Settings** from the **Options** menu.

14 On the **Axis** page set:

x min	2e-3
x max	6e-3
y min	-1e-3
y max	1e-3

15 Click **OK**; Figure 5-17 results.

PHYSICS SETTINGS—MODE B

Now switch the electric field to Mode B as described in Figure 5-14.

Boundary Conditions

1 Select **2 Conductive Media DC (emdc)** from the **Multiphysics** menu.

2 Open the **Boundary Settings** dialog box from the **Physics** menu.

3 Enter the boundary conditions from the following table, then click **OK**:

BOUNDARY	1	5	9	12	2-4, 6-8, 10, 11, 13-16
Boundary condition	Electric potential	Electric potential	Electric potential	Electric potential	Electric insulation
V_0	114	193	0	79	

COMPUTING THE SOLUTION—MODE B

Click the **Solve** button on the Main toolbar.

POSTPROCESSING AND VISUALIZATION—MODE B

1 Click **Zoom Extents**.

2 To create Figure 5-18 do exactly as for Figure 5-16.

When making Figure 5-19, follow the instructions for Figure 5-17 except that the **x** and **y** edit fields should be `linspace(2.5e-3,2.75e-3,10)` and `linspace(2e-3,2e-3,10)`, respectively

6

Chemical Engineering Module 3.2b Model Library Update

This chapter contains an updated model for the Chemical Engineering Module 3.2b Model Library:

- Rising Bubble Modeled with the Level Set Method

Rising Bubble Modeled with the Level Set Method

This model shows the concept of modeling two-phase flow using the level set method. This method is well suited for problems with moving boundaries in which the topology of the geometry changes with time. A rising bubble that travels up through a heavier fluid and finally merges with the light fluid at the top causes this kind of topology change. For problems where the topology is unchanged as a function of time, as in free surface movement in a tank (no splashing) and impeller stirring, the ALE (arbitrary Lagrangian-Eulerian) method is preferred.

In this specific problem, three different regions exist initially: the bubble, the light fluid at top of the container, and the heavy fluid surrounding the bubble (see Figure 6-1).

Introduction

This model treats a bubble of a light fluid traveling through a heavier fluid to finally merge with the lighter fluid already residing at the top of the container. The initial positions of the two phases are shown in Figure 6-1 below.

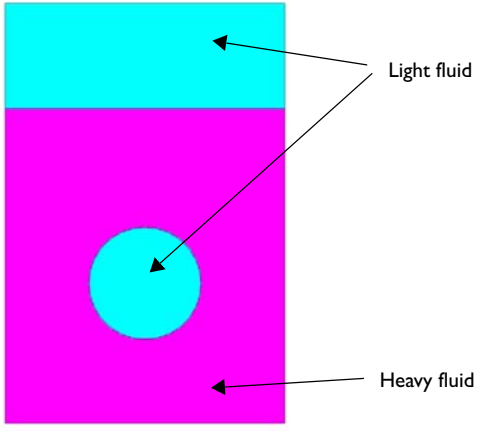


Figure 6-1: Initial position of the drop.

The driving force for the movement of the bubble is the difference in density between the two phases. The bubble of lighter fluid is subjected to a relative force in the positive y-direction. The bubble therefore rises through the heavy fluid.

The basic concept of modeling two-phase flow with the level set method is characterized by a detailed description of the interface between the two fluids by tracing the level curve of an auxiliary function. The *level set function* ϕ behaves so that the interface is defined where $\phi = 0$. The heavy fluid is located in the domain where $\phi > 0$, while the lighter fluid resides the domain where $\phi < 0$.

Model Definition—Fluid Flow

The fluid flow is governed by the Navier-Stokes equations:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\right) - \nabla \cdot [\mu(\nabla \mathbf{u} + \nabla \mathbf{u}^T)] + \nabla p = \sigma \kappa \mathbf{n} \delta + \rho \mathbf{g} \tag{6-1}$$

$$\nabla \cdot \mathbf{u} = 0 \tag{6-2}$$

In the above equations, ρ (kg/m^3) denotes the density, $\mathbf{u}$ the velocity vector (m/s), t time (s), μ the viscosity (Pa s), p the pressure (Pa), σ the surface tension coefficient, κ the surface curvature at the fluid interface, $\mathbf{n}$ is the unit normal to the interface drawn outwards from the light fluid into the heavier one, δ is the delta function which has unit integral and has its maximum at the interface between the fluids and zero otherwise, and $\mathbf{g}$ the is the gravitation vector.

In Equation 6-1, the term $\sigma\kappa\mathbf{n}\delta$ defines the surface tension forces, and $\rho\mathbf{g}$ defines the gravity force. Equation 6-2 is the continuity equation, which states that mass is conserved.

INITIAL CONDITION

Initially, at $t = 0$, the velocity and pressure are both set to zero.

BOUNDARY SETTINGS

No-slip conditions, $\mathbf{u} = 0$, are used everywhere.

SURFACE TENSION

To avoid computing the curvature explicitly in the surface tension force, you can integrate this force by parts (after multiplying with the test functions) using a surface divergence theorem. By doing so, the force can be applied by adding the weak contribution

$$\sum_{i=1}^{d} \sigma\delta(\nabla_s \hat{\mathbf{u}}_i)_i$$

where $\nabla_s = (\mathbf{I} - \mathbf{n}\mathbf{n}^T) \cdot \nabla$ represents the surface gradient operator and $\hat{\mathbf{u}}_i$ the test function for the ith velocity component. This way of computing the surface tension force is used below and results in better accuracy than the explicit application of the force, because it only uses first-order derivatives (of the unknowns).

Model Definition—Level Set Equation

The level set method models the fluid interface by tracing the isolines of a dimensionless level set function, ϕ. The zero level set at $\phi = 0$ determines the position of the interface. ϕ is transported by the following equation:

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0 \tag{6-3}$$

The geometric properties of the interface are easily determined from the level set function. The surface curvature at the fluid interface, κ, and the unit normal to the interface, $\mathbf{n}$, are defined as:

$$\kappa = \nabla \cdot \left. \frac{\nabla \phi}{|\nabla \phi|} \right|_{\phi = 0} \tag{6-4}$$

$$\mathbf{n} = \left. \frac{\nabla \phi}{|\nabla \phi|} \right|_{\phi = 0} . \tag{6-5}$$

INITIAL CONDITION

The initial condition for the level set function is the key to the whole model. You define a function so that the level set function is zero at the interface between the two fluids, greater than zero for the heavy fluid, and less than zero at the light fluid. The function

in this example is: `min((2.25-y),sqrt(x^2+(y-1)^2)-0.4)`. Figure 6-2 shows a visualization of this function.

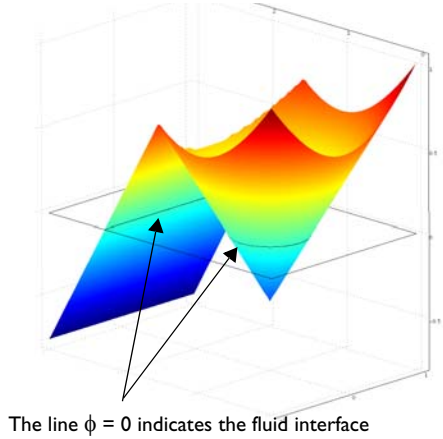


Figure 6-2: Initial shape of the level set function.

BOUNDARY SETTINGS

For the level set function, all boundaries are defined as insulating boundaries.

Results and Discussion

Figure 6-3 contains snapshots of the velocity field and the contour $\phi = 0$. The snapshots show how the bubble travels up through the heavy fluid and merges with the light fluid above. The sequence also shows how the shape of the bubble changes as it travels upward in the container.

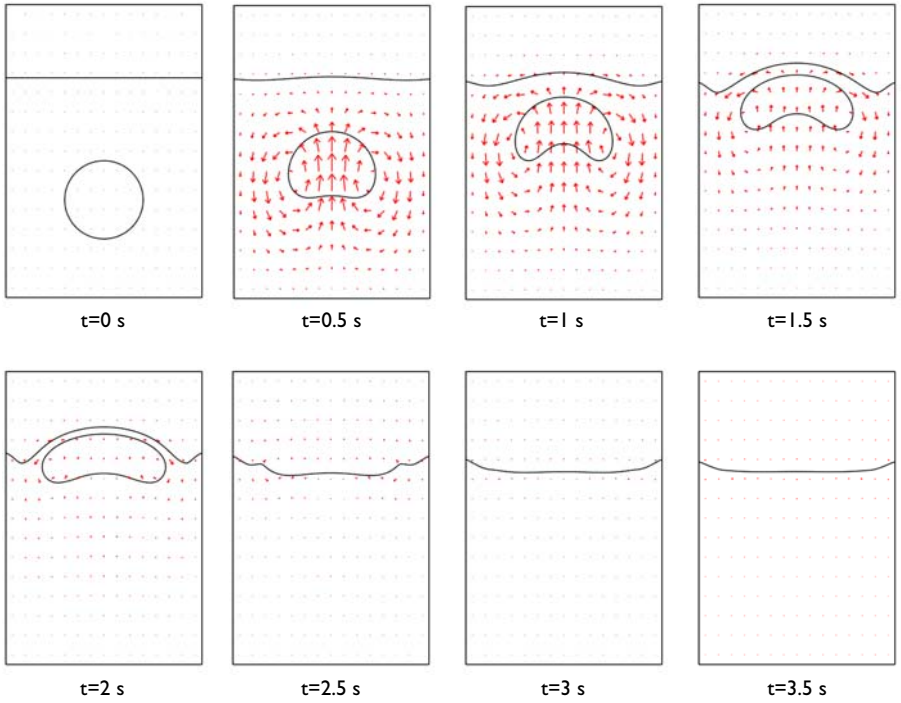


Figure 6-3: Snapshot images showing the velocity field (arrows) and the interface between the two phases (solid black line).

One important factor to monitor throughout a level set simulation is the total mass. The integral of the mass over the domain as a function of time gives a clear picture of the mass conservation. Figure 6-4 shows the dimensionless total mass in the domain over time. At the final time step, t = 5 s, 99.5% of the mass remains accounted for, which is a pretty decent result. You can obtain an even better conservation of mass by refining the mesh.

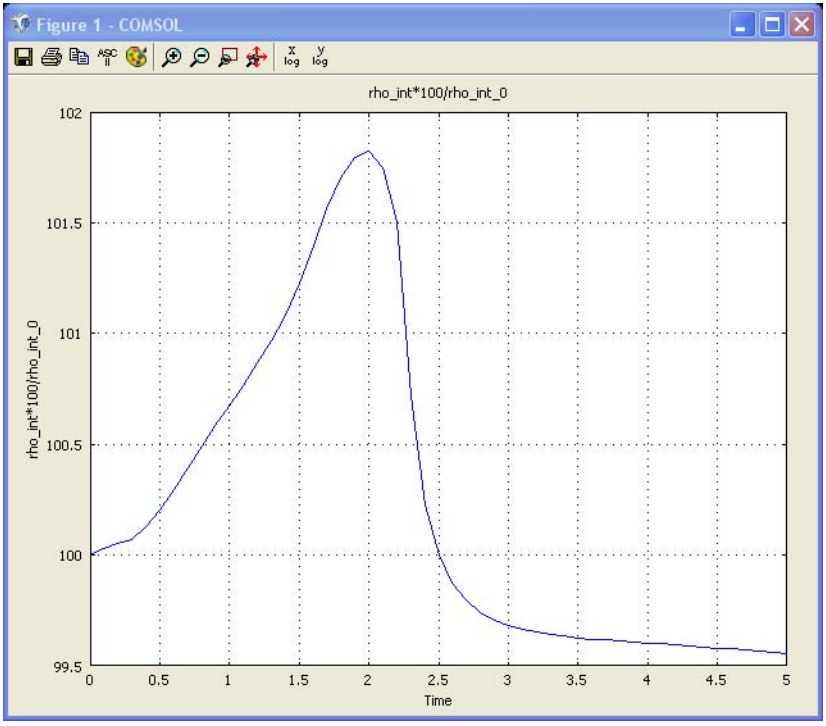


Figure 6-4: Total mass in the domain as a function of time. At the end of the simulation, about 99.5% of the mass remains.

Modeling in COMSOL Multiphysics

An important aspect in this model is the treatment of the varying density as a function of the level set function. Using density ρ_1 in the lighter fluid and density ρ_2 in the heavier fluid, you can express a total density function according to

$$\rho = \rho_1 + H(\phi) \cdot (\rho_2 - \rho_1). \tag{6-6}$$

$H(\phi)$ in Equation 6-6 is a step function defined according to Equation 6-7:

$$H(\phi) = \begin{cases} 0 \; if \; \phi < 0 \\ 1 \; if \; \phi > 0 \end{cases} \tag{6-7}$$

Instead of using a step function with infinite derivative, this example uses the built-in smooth step functions in COMSOL Multiphysics. The function `flc2hs` is a smooth, C^2-continuous Heaviside function, and you define it as `flc2hs(arg,h_scale)` where `arg` is the level set function ϕ, scaled with its gradient, and `h_scale` is a mesh-dependent scale for the resolution of the interface.

Modeling Using the Graphical User Interface

MODEL NAVIGATOR

Using the Reacting Flow predefined multiphysics coupling, you get the application modes that you need and the predefined coupling of the velocities from the Incompressible Navier-Stokes application mode to the convective transport of the level set function.

1. In the **Model Navigator**, select **Predefined Multiphysics Couplings> Fluid-Chemical Reactions Interaction>Reacting Flow>Transient analysis** from the list of application modes.
2. Change the entry in the **Dependent variables** edit field to `u v p phi`.
3. Click **Add** and then **OK** to close the Model Navigator.

GEOMETRY MODELING

1. Shift-click the **Rectangle/Square** button in the Draw toolbar.
2. Specify the rectangle size according to the table below.

PROPERTY	EXPRESSION
Width	2
Height	3
Position, x	-1
Position, y	0

3. Click **OK** and then click the **Zoom Extents** button in the Main toolbar.

OPTIONS AND SETTINGS

1 Enter the following constant names and expressions in the **Constants** dialog box. The descriptions are optional.

NAME	EXPRESSION	DESCRIPTION
eta	1	viscosity
rho1	1	density of light fluid
rho2	10	density of heavy fluid
gy	-10	gravity constant
sigma	1	surface tension coeff.

2 Click **OK** to close the dialog box.

3 Enter the following scalar expressions in the **Scalar Expressions** dialog box that you open by selecting **Expressions>Scalar Expressions** in the **Options** menu.

NAME	EXPRESSION	DESCRIPTION
H	flc2hs(arg,h_scale)	step function switching at phi=0
DELTA	fldc2hs(arg,h_scale)	delta function at the phase interface
arg	phi/gphi	argument of H and DELTA
gphi	sqrt(phix^2+phiy^2)	size of the gradient of phi
rho	rho1+H*(rho2-rho1)	density
Fy	gy*rho	volume force in y-direction
phi_0	min((2.25-y),sqrt(x^2+(y-1)^2)-0.4)	initial phi
H_0	flc2hs(phi_0,h_scale)	initial H
rho_0	rho1+H_0*(rho2-rho1)	initial rho
h_scale	1.5*h	scale for resolution of interface
d_add	2*h^2	artificial diffusion at interface

4 Click **OK** to close the dialog box.

5 Select subdomain 1. Enter the following subdomain expressions in the **Subdomain Expressions** dialog box, which you open by selecting **Expressions>Subdomain Expressions** in the **Options** menu.

NAME	EXPRESSION
nx	phix/gphi
ny	phiy/gphi

6 Click **OK** to close the dialog box.

7 Enter the following integration coupling variables in the **Subdomain Integration Variables** dialog box, which you open by selecting **Options>Integration Coupling Variables>Subdomain Variables**:

NAME	EXPRESSION	INTEGRATION ORDER	GLOBAL DESTINATION
rho_int	rho	4	yes
rho_0_int	rho_0	4	yes

8 Click **OK** to close the dialog box.

PHYSICS SETTINGS—INCOMPRESSIBLE NAVIER-STOKES

Select **1 Incompressible Navier-Stokes (chns)** from the **Multiphysics** menu.

Boundary Conditions

1 Open the **Boundary Conditions** dialog box from the **Physics** menu.

2 Select all boundaries and set the **No slip** boundary condition (this is the default boundary condition for Incompressible Navier-Stokes).

3 Click **OK** to close the dialog box.

Subdomain Settings

1 Open the **Subdomain Settings** dialog box from the **Physics** menu.

2 Enter subdomain settings according to the table below

SUBDOMAIN	1
Density	rho
Dynamic viscosity	eta
Volume force x-dir	0
Volume force x-dir	Fy

3 Click **OK**.

To apply the surface tension force, add a weak expression from the discussion above:

4 Open the **Equation System>Subdomain Settings** dialog box from the **Physics** menu.

5 Click the **Weak** tab and enter weak complement settings according to the following table:

SUBDOMAIN	I
weak	`(-ny*ny*test(ux)+nx*ny*(test(uy)+test(vx))-nx*nx*test(vy))*sigma*DELTA 0 0 0`

6 Click **OK**.

Point Settings

1 Open the **Point Settings** dialog box from the **Physics** menu.

2 Select point 3 from the **Point selection** list (the lower right corner).

3 Select the **Point constraint** check box and leave $\mathbf{p}_0$ at zero.

4 Click **OK**.

PHYSICS SETTINGS—CONVECTION AND DIFFUSION

Select **2 Convection and Diffusion (chcd)** from the **Multiphysics** menu.

Boundary Conditions

1 Open the **Boundary Conditions** dialog box from the **Physics** menu.

2 Select all boundaries and set the **Insulation/Symmetry** boundary condition (this is the default boundary condition for Convection and Diffusion application mode).

3 Click **OK** to close the dialog box.

Subdomain Settings

1 Open the **Subdomain Settings** dialog box from the **Physics** menu.

2 Enter the diffusion coefficient according to the table below (all other settings are correct by default):

SUBDOMAIN	I
Diffusion coefficient	`d_add*h_scale*DELTA`

3 Click the **Artificial Diffusion** button.

4 Select the **Streamline Diffusion** check box and then select **Petrov-Galerkin/ Compensated** from the list.

5 Click **OK** to close the **Artificial Diffusion** dialog box.

6 Go to the **Init** tab and enter `phi_0` as initial value.

7 Click **OK** to close the **Subdomain Settings** dialog box.

MESH GENERATION

Use a mapped mesh in this model:

1 Select **Map Mesh** from the **Mesh** menu.

2 Click the **Boundary** tab and select boundary 1.

3 Select the **Constrained edge element distribution** check box and enter `75` in the **Number of mesh elements** edit field.

4 Repeat step 3 above for boundary 2 but with 50 mesh elements.

5 Click **Remesh** and then click **OK**.

COMPUTING THE SOLUTION

1 Open the **Solver Parameters** dialog box.

2 Make sure that the **Time dependent** solver is selected.

3 In the **Time stepping** area, type `0:0.1:5` in the **Times** edit field.

4 Set **Relative tolerance** to `0.001` and the **Absolute tolerance** to `0.0001`.

5 Click **OK** to close the **Solver Parameters** dialog box.

6 Click the **Solve** button in the Main toolbar to compute the solution.

POSTPROCESSING AND VISUALIZATION

To create the images in Figure 6-3 on page 166, do the following steps:

1 Open the **Plot Parameters** window from the **Postprocessing** menu.

2 In the **Plot type** frame on the **General** page, select **Contour**, **Arrows** and **Geometry edges**. Clear all other plot types.

3 Clear the **Element refinement Auto** check box and enter `5` in the **Element refinement** edit field.

4 Select `0` from the **Solution at time** drop down menu.

5 Go to the **Contour** page and select `phi` as **Contour data**.

6 In the **Contour levels** frame, select **Vector with isolevels** and enter `0` in the edit field.

7 Select black as **Contour color**.

8 Click the **Arrows** tab.

9 Select **Velocity field (ns)** in the **Predefined quantities** list on the **Subdomain data** tab.

10 Click **OK**.

To create the snapshots at other time steps than 0 s, use the same plot parameters as above but for other values from the **Solution at time** list. The plots in Figure 6-3 uses a thicker line width (2.0) than the default settings and manual scaling of the arrows to account for the varying velocity.

To create the plot in Figure 6-4 on page 167, do the following steps:

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

2 Select all time steps in **Solutions to use** by pressing Ctrl+A.

3 Click the **Point** tab and enter `rho_int/rho_0_int*100` in the **Expression** edit field.

4 Click **OK**.

7

Trusses in Structural Mechanics Module 3.2b

Trusses are elements which can only sustain axial forces. They have displacements as degrees of freedom. Trusses are sometimes referred to as bars or spars. They live on boundaries in 2D and edges in 3D. The truss application modes support the same analysis types as the continuum application modes. You can use trusses to model truss works where the edges are straight but also to model sagging cables like the deformation of a wire exposed to gravity. In such applications trusses are often referred to as cable elements.

This chapter describes the application modes for in-plane and 3D trusses in the Structural Mechanics Module 3.2b.

Theory Background

Trusses is modeled using Lagrange shape function. The Lagrange shape function makes it possible to specify both normal strains and Green-Lagrange strains to handle small strains as well as large deformations.

Strain-Displacement Relation

The axial strain ε_n is calculated by expressing the global strains in tangential derivatives and projecting the global strains on the edge.

$$\varepsilon_n = \mathbf{t}^t \varepsilon_{gT} \mathbf{t} \tag{7-1}$$

where $\mathbf{t}$ is the edge tangent vector and ε_{gT} is defined as

$$\varepsilon_{gT} = \begin{bmatrix} \varepsilon_{xT} & \varepsilon_{xyT} & \varepsilon_{xzT} \\ \varepsilon_{xyT} & \varepsilon_{yT} & \varepsilon_{yzT} \\ \varepsilon_{xzT} & \varepsilon_{yzT} & \varepsilon_{zT} \end{bmatrix} \tag{7-2}$$

The strains can be expressed as either engineering strains for small displacements or Green strains for large displacements. The Green strain tensor used for large displacements is defined as

$$\varepsilon_{ijT} = \frac{1}{2}\left(\left.\frac{\partial u_i}{\partial x_j}\right|_T + \left.\frac{\partial u_j}{\partial x_i}\right|_T + \left.\frac{\partial u_k}{\partial x_i}\right|_T \cdot \left.\frac{\partial u_k}{\partial x_j}\right|_T \right) \tag{7-3}$$

The engineering strain tensor used for small displacements is defined as

$$\varepsilon_{ijT} = \frac{1}{2}\left(\left.\frac{\partial u_i}{\partial x_j}\right|_T + \left.\frac{\partial u_j}{\partial x_i}\right|_T \right) \tag{7-4}$$

The axial strain written out becomes

$$\begin{aligned} \varepsilon_n = {} & t_x(\varepsilon_{xT} t_x + \varepsilon_{xyT} t_y + \varepsilon_{xzT} t_z) + \\ & t_y(\varepsilon_{xyT} t_x + \varepsilon_{yT} t_y + \varepsilon_{yzT} t_z) + \\ & t_z(\varepsilon_{xzT} t_x + \varepsilon_{yzT} t_y + \varepsilon_{zT} t_z) \end{aligned} \tag{7-5}$$

Stress-Strain Relation

The constitutive relation for the axial stress including thermal strain and initial stress and strain is

$$\sigma_n = E(\varepsilon_n - \alpha(T - T_{ref}) - \varepsilon_{ni}) + \sigma_{ni} \tag{7-6}$$

Implementation

Using the principle of virtual work results in the following weak formulation

$$dW = d\int_V (-\varepsilon_n \sigma_n + \mathbf{u}^t \mathbf{F}_V) dV + \sum_i \mathbf{u}^t \mathbf{F}_{Pi} \tag{7-7}$$

where the summation stands for summation over all points in the geometry. Replacing the integration over the cross section with the cross-sectional area (A) and the volume forces with line forces, the equation becomes

$$dW = \int_L (-\varepsilon_{ntest} \sigma_n A + \mathbf{u}^t_{test} \mathbf{F}_L) dL + \sum_i \mathbf{u}^t_{test} \mathbf{F}_{Pi} \tag{7-8}$$

Straight Edge Option

The optional constraint to enforce the nodes to lie on the straight line between the end points of the edge are formulated as follows:

Starting with the large displacement case, let $\mathbf{x}_{d1}$ and $\mathbf{x}_{d2}$ be the deformed position of the two end points of the edge

$$\mathbf{x}_{di} = \mathbf{u}_i + \mathbf{x}_i \tag{7-9}$$

where $\mathbf{u}_i$ is the displacement, and $\mathbf{x}_i$ is the coordinate (undeformed position) at end point i. The equation for the straight line through the end points is

$$\mathbf{x} + \mathbf{u} = \mathbf{x}_{d1} + t\mathbf{a} \tag{7-10}$$

where t is a parameter along the line, and $\mathbf{a}$ is the direction vector for the line. $\mathbf{a}$ is calculated from the deformed position of the end points as

$$\mathbf{a} = \mathbf{x}_{d2} - \mathbf{x}_{d1} \tag{7-11}$$

The constraints for the edge is derived by substituting the parameter t from one of the scalar equations in Equation 7-10 into the remaining ones. In 2D the constraint equations become

$$(x + u - x_{d1})a_y - (y + v - y_{d1})a_x \tag{7-12}$$

In 3D the two constraints equations become

$$\begin{aligned}(x + u - x_{d1})a_z - (z + w - z_{d1})a_x \\ (y + v - y_{d1})a_z - (z + w - z_{d1})a_y\end{aligned} \tag{7-13}$$

To avoid problems when the edge is directed in one of the coordinate axes directions, a third constraint is added. This constraint is a linear combination of the two earlier constraints:

$$(y + v - y_{d1})a_x - (x + u - x_{d1})a_y$$

You need a linear constraint in order for the solution of the small displacement problem to become independent of the solver. The linear relation for the displacement is

$$\mathbf{u} = \frac{\mathbf{u}_1(x_{n2} - x_n) + \mathbf{u}_2(x_n - x_{n1})}{(x_{n2} - x_{n1})} + u_{ax}(\mathbf{x}_2 - \mathbf{x}_1) \tag{7-14}$$

where u_{ax} is the axial displacement along the edge, and x_n are a linear parameter along the edge

$$x_n = \frac{x(x_2 - x_1) + y(y_2 - y_1) + z(z_2 - z_1)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}}$$

Eliminating u_{ax} fromEquation 7-14 results in the following linear constraint in 2D

$$\left[\frac{u_1(x_{n2} - x_n) + u_2(x_n - x_{n1})}{(x_{n2} - x_{n1})} - u\right](y_2 - y_1) - \left[\frac{v_1(x_{n2} - x_n) + v_2(x_n - x_{n1})}{(x_{n2} - x_{n1})} - v\right](x_2 - x_1)$$

and the following three linear constraints in 3D:

$$\left[\frac{u_1(x_{n2}-x_n)+u_2(x_n-x_{n1})}{(x_{n2}-x_{n1})}-u\right](z_2-z_1)-\left[\frac{w_1(x_{n2}-x_n)+w_2(p-x_{n1})}{(x_{n2}-x_{n1})}-w\right](x_2-x_1)$$

$$\left[\frac{v_1(x_{n2}-x_n)+v_2(x_n-x_{n1})}{(x_{n2}-x_{n1})}-v\right](z_2-z_1)-\left[\frac{w_1(x_{n2}-x_n)+w_2(x_n-x_{n1})}{(x_{n2}-x_{n1})}-w\right](y_2-y_1)$$

$$\left[\frac{v_1(x_{n2}-x_n)+v_2(x_n-x_{n1})}{(x_{n2}-x_{n1})}-v\right](x_2-x_1)-\left[\frac{u_1(x_{n2}-x_n)+u_2(x_n-x_{n1})}{(x_{n2}-x_{n1})}-u\right](y_2-y_1)$$

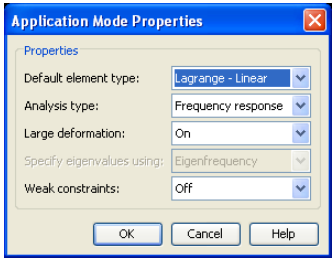
Application Mode Description

This section describes how to define a truss model. It is divided into the following sections:

- Properties
- Scalar Variables
- Material
- Cross Section
- Constraint
- Load
- Thermal Coupling
- Initial Stress and Strain

Properties

To open the **Application Mode Properties** dialog box, choose **Properties** from the **Physics** menu.



In the **Application Mode Properties** dialog box you control different global settings for the model.

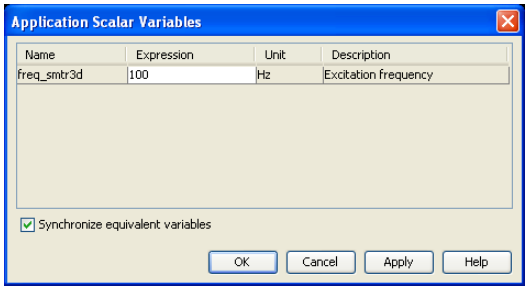
- **Default element type**: The selected finite element type that makes up the discretized finite element model is the default on all new boundaries/edges, and the choice does not affect boundaries/edges already created. Available elements are:
  - **Lagrange - Linear**
  - **Lagrange - Quadratic**
  - **Lagrange - Cubic**
  - **Lagrange - Quartic**
  - **Lagrange - Quintic**
- **Analysis type**: A list of different analyses to perform. It affects both the equations and what solver to use through the **Auto select solver** option in the **Solver Parameters** dialog box. The available analysis types use the following solvers:

ANALYSIS TYPE	COMSOL MULTIPHYSICS SOLVER
Static linear	Stationary linear
Static nonlinear	Stationary nonlinear
Eigenfrequency	Eigenvalue
Time dependent	Time dependent
Frequency response	Parametric
Parametric	Parametric
Quasi-static transient	Time dependent
Linear Buckling	Eigenvalue

- **Large deformation**: This list controls whether or not the model should support large deformations.
- **Specify eigenvalues using**: This list controls how to work with eigenmode analyses. Here you should specify **Eigenvalue** or **Eigenfrequency/Critical load factor**; this property is enabled only for eigenfrequency and linear-buckling analyses.
- **Weak constraints**: Controls if weak constraints should be available or not. Available options are **Off**, **Ideal**, and **Non-ideal**. Use weak constraints for accurate reaction force computation, solving for the reaction force. When weak constraints are enabled, all constraints are weak constraints by default, but it is possible to control this for individual domains.

Scalar Variables

Scalar variables are only used for frequency response analysis. The **Scalar Variables** menu item on the **Physics** menu is enabled only when **Frequency Response** is selected as **Analysis Type** in the **Application Mode Properties** dialog box.



The In-Plane Truss and 3D Truss application modes only have one scalar variable.

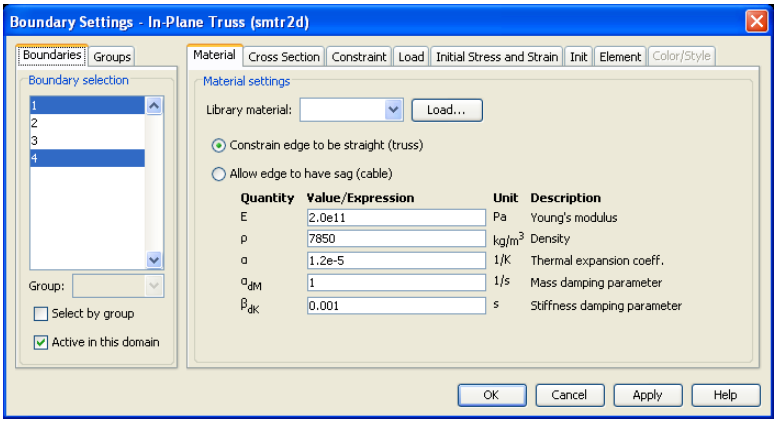
PROPERTY	VARIABLE	DESCRIPTION
f	`freq_smtr2d, freq_smtr3d`	Excitation frequency

The excitation frequency is the frequency of the harmonic loads/constraints in a frequency response analysis.

When **Frequency response** is selected as analysis type, the default solver is the parametric solver, making it easy to perform a frequency sweep over several excitation frequencies in one analysis. In this case, enter `freq_smtr2d` as the **Name of parameter** on the **Parametric** page in the **Solver Parameters** dialog box. Doing so makes the values entered in the **List of parameter values** override the excitation frequency entered in the **Application Scalar Variables** dialog box.

Material

The material properties are defined on the **Material** page in the **Boundary Settings** dialog box for the In-Plane Truss and in the **Edge Settings** dialog box for the 3D Truss.



The material properties are shown in the table below.

PARAMETER	VARIABLE	DESCRIPTION	COMMENT
E	E	Young's modulus	
ρ	rho	Density	
α	alpha	Thermal expansion coefficient	
α_{dM}	alphadM	Mass damping parameter	
β_{dK}	betadK	Stiffness damping parameter	

Young's modulus Defines the modulus of elasticity, E of the material. It is the spring stiffness in Hooke's law, shown below in 1D form

$$\sigma = E\varepsilon$$

Density This material property, ρ, specifies the density of the material.

Thermal expansion coefficient Defines how much a material expands due to an increase in temperature.

$$\varepsilon_{th} = \alpha(T - T_{ref})$$

where ε_{th} is the thermal strain, T is the strain temperature and T_{ref} is the stress free reference temperature.

Mass damping parameter Defines the Rayleigh damping models mass damping, α_{dM}.

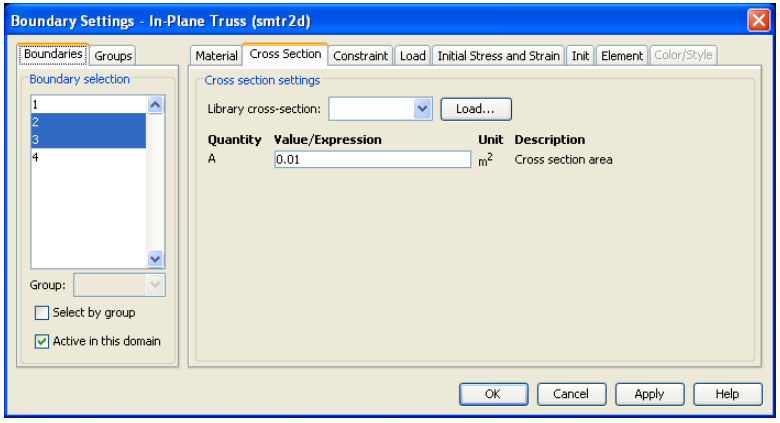
Stiffness damping parameter Defines the Rayleigh damping models stiffness damping, β_{dK}.

The **Constrain edge to be straight (truss)** and **Allow edge to have sag (cable)** buttons control the addition of an additional constraint, forcing the edge to be straight. The default is to add the constraint. Using this additional constraint removes the need to use a mesh with only one element per edge. The problem with internal nodes is that they makes the problem singular, because the truss only has stiffness in the axial direction. The same applies when using higher-order elements. The additional constraint increases the solution time, especially for large 3D and transient problems. The remedy to this is to turn off the constraint option (click the **Allow edge to have sag (cable)** button) and use linear elements together with a very coarse mesh consisting of only one element/edge.

For problems where you want to model the sag and do not have a straight line between the edge points, click the **Allow edge to have sag (cable)** button and use that setting together with the **Large deformation** option and a suitable mesh with internal nodes.

Cross-Section Properties

You define cross-sectional properties on the **Cross Section** page in the **Edge/Boundary** settings dialog box.



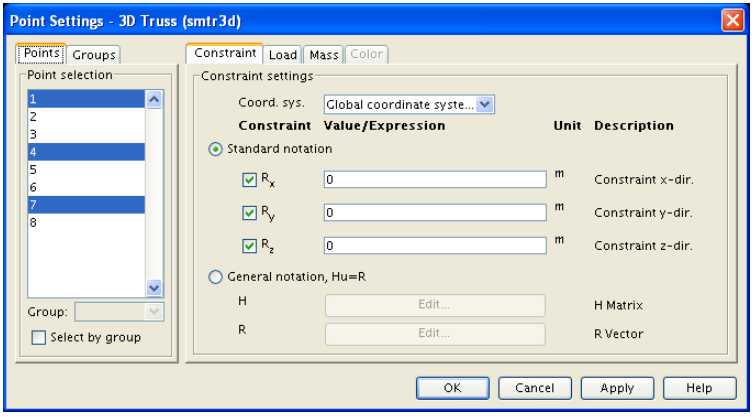
The only cross-section property in these application modes is the cross-section area:

PARAMETER	VARIABLE	DESCRIPTION	COMMENT
A	A	Cross-section area	

Constraints

A constraint specifies the displacement of a certain part of the truss. Constraints can be defined on all valid domain levels such as points, edges/boundaries. You control the constraints from the **Constraint** page in the **Boundary Settings**, **Edge Settings**, and **Point Settings** dialog boxes.

The following figure shows the **Point Settings** dialog box for the 3D Truss application mode, but the page looks similar on all domain levels in both truss application modes.



An example of a truss Constraint page, taken here from the 3D Truss application mode Point Settings dialog box.

The **Coordinate system** list lets you control in which coordinate system you want the constraint defined. Available options are:

- Global coordinate system
- Tangential and normal coordinate system, only available on boundaries for the in-plane truss.
- User-defined coordinate systems, if there are any local coordinate systems defined. Read more about creation of coordinate system in the section "Coordinate Systems" on page 59.

You can prescribe a constraint in two ways:

- In standard notation you constrain each displacement direction independently. The check box in front of $\mathbf{R}_x$, $\mathbf{R}_y$, and $\mathbf{R}_z$ activates the constraint, and you can then enter the value or expression for the displacement in the corresponding edit fields. The default value is `0` (no displacement).
- In general notation, the H matrix and R vector in the relation

$$Hu = R$$

lets you specify constrain as any linear combination of the available variables.

For the In-Plane Truss application mode the relation is

$$H\begin{bmatrix} u \\ v \end{bmatrix} = R$$

For the 3D Truss application mode the relation is

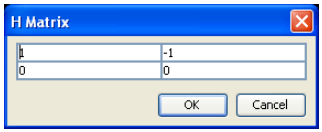
$$H\begin{bmatrix} u \\ v \\ w \end{bmatrix} = R$$

To enter the H matrix and the R vector, use special matrix dialog boxes that you open by clicking the respective **Edit** buttons. For example, you can achieve the condition $u = v$ in the In-Plane Truss application mode using the settings

$$H = \begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix} \qquad R = \begin{bmatrix} 0 \\ 0 \end{bmatrix},$$

which force the domain to move only diagonally in the x-y plane.

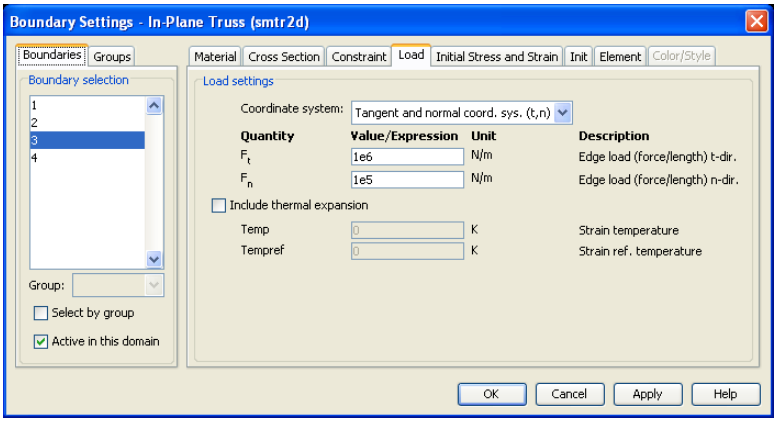
The **H Matrix** dialog box for this example is



Loads

A load is a general name for all forces applied to the structure. You can specify loads on all domain types using the **Load** page in the **Boundary Settings, Edge Settings**, and **Point Settings** dialog boxes. The following picture shows the **Boundary Settings** dialog

box for the In-Plane Truss application mode, but the page looks similar on all domain levels.



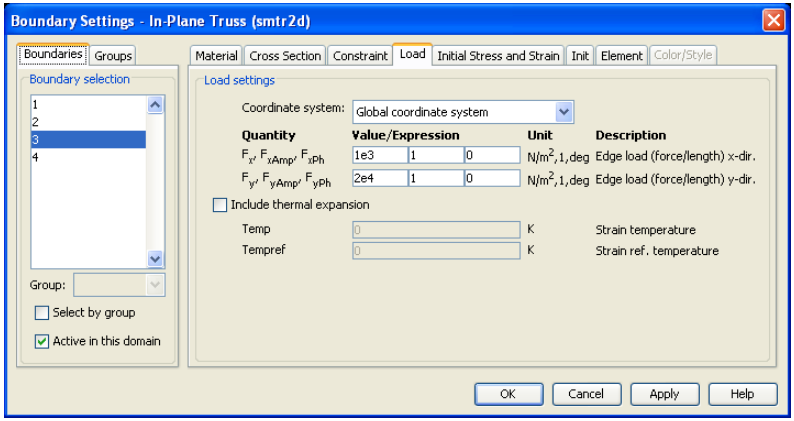
The loads are defined in the following way. The SI unit is shown in parenthesis.

POINT	EDGE, BOUNDARY
force (N)	force/length (N/m)

With the **Coordinate system** list you control in what coordinate system the load is defined. Available options are:

- Global coordinate system
- Tangential and normal coordinate system, only available on boundaries for the in-plane truss.
- User-defined coordinate systems, if there are any local coordinate systems defined. Read more about creation of coordinate system in the section “Coordinate Systems” on page 59.

For the frequency response analysis type, you need to specify additional input data. The analysis type is controlled from the **Application Mode Properties** dialog box. When frequency response is selected as analysis type, the **Load** page changes appearance to



For frequency response analysis the harmonic load is split into 3 different parameters:

- Value (F)
- Amplitude (F_{amp})
- Phase (F_{Ph})

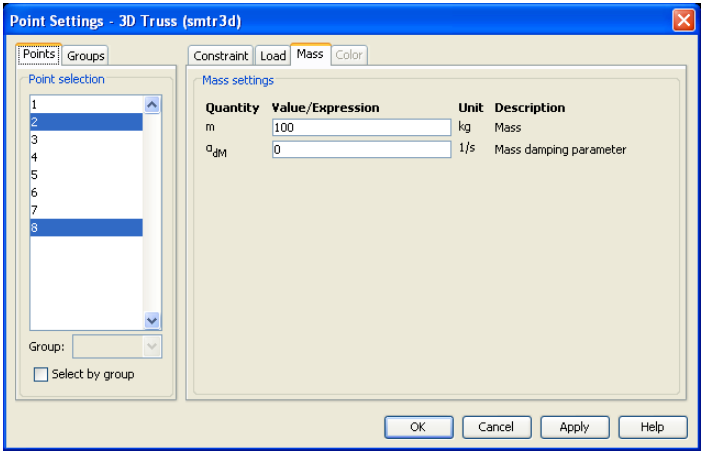
Together they define a harmonic load whose amplitude and phase shift can vary with the excitation frequency f.

$$F_{freq} = F \cdot F_{Amp}(f) \cdot \cos(2\pi f + F_{Ph}(f))$$

On the edge and boundary domain level additional options are available controlling if and how thermal strains should be included in the analysis. They are explained in the section “Thermal Coupling” on page 189.

Discrete Mass

Discrete mass is concentrated to a point in contrast to distributed mass modeled through the density and area of the truss. You specify discrete mass on the **Mass** page in the **Point Settings** dialog box.



The mass properties are shown in the table below.

PARAMETER	VARIABLE	DESCRIPTION	SI UNITS	COMMENT
m	m	Mass	kg	
α_{dM}	alphadM	Mass damping parameter	1/s	

Thermal Coupling

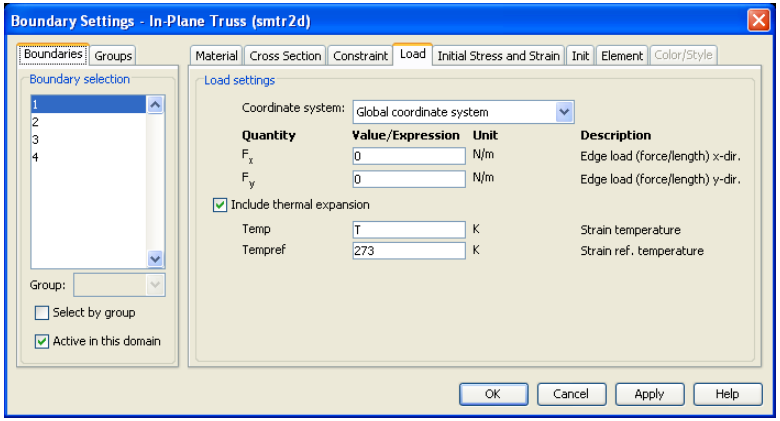
Material expands with temperature, which causes thermal strains to develop in the material. The trusses can handle any temperature variation along the truss. The thermal strains together with the initial strains and elastic strains from structural loads form the total strain.

$$\varepsilon = \varepsilon_{el} + \varepsilon_{th} + \varepsilon_i$$

where

$$\varepsilon_{th} = \alpha(T - T_{ref})$$

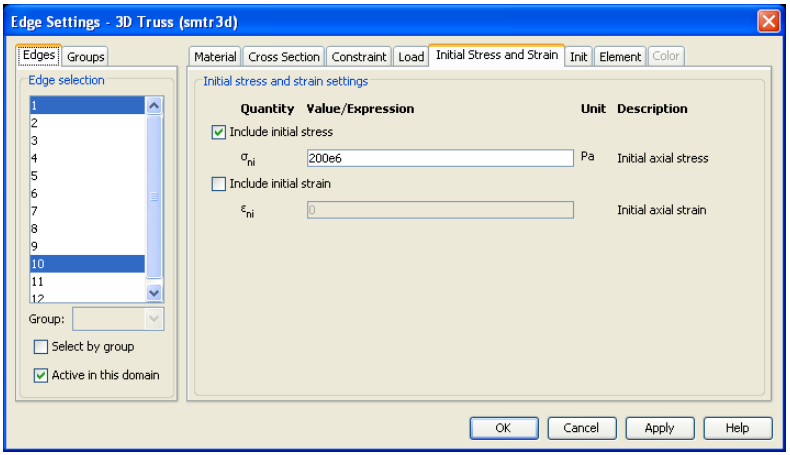
Thermal coupling means that the thermal expansion is included in the analysis. Details on thermal coupling is found on page 177. Thermal effects are specified on the **Load** page in the **Edge/Boundary Settings** dialog box.



The **Include thermal expansion** check box adds thermal effects. In the **Temp** and **Tempref** edit fields you specify the strain temperature T and stress free reference temperature T_{ref}, respectively. Use the **Material** page to define the thermal expansion coefficient (described in "Material" on page 183). T and T_{ref} can be any expression and can be a dependent variable for temperature from another application modes solving the heat transfer problem. The temperature coupling can be used in any type of analysis.

Initial Stress and Strain

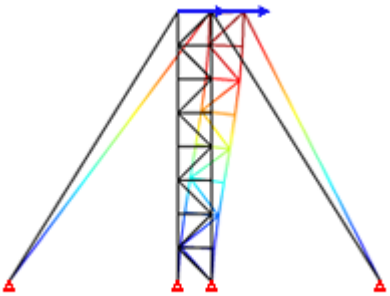
Initial stress and strain can be included in the analysis. Initial stress and strain are specified on the **Initial Stress and Strain** page in the **Edge/Boundary Settings** dialog box.



The option to include initial stresses and strains is controlled independently using the two check boxes **Include initial stress** and **Include initial strain**.

In-Plane Truss Application Mode

This section describes the In-Plane Truss application mode in the Structural Mechanics Module to analyze planar lattice trusses or sagging cable-like structures.



In-Plane Truss application modes are defined on edges in 2D. All settings for the application mode are described in "Application Mode Description" on page 180.

Variables and Space Dimensions

The degrees of freedom (dependent variables) are the global displacements u and v in the global x and y directions, respectively.

Variables

A large number of variables are available for use in expressions and for postprocessing purposes. In addition to the variables listed below, almost all application mode parameters are available as variables. Some variables are different for different analyses, which you can see in the Analysis column. For frequency response analysis, a number of new variables are available (see Table 7-1). In addition to variables such as stress and strain, you can also access variables for the amplitude and phase of those variables using the convention of adding `_amp` or `_ph` after the name of the variable, for example:

- `en_amp` is the amplitude of the axial strain.
- `sn_ph` is the phase of the axial stress.

Table 7-1 uses an index convention where a single index on u_i/u_i means that i runs over the global displacements (u,v). The Analysis column uses the following abbreviations:

ANALYSIS	ABBREVIATION
Static	S
Frequency response	F
Time dependent	T
Eigenfrequency	E

TABLE 7-1: IN-PLANE TRUSS APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
u_i	u_i	All	All	x_i displacement	u_i
u_it	u_{it}	T	All	x_i velocity	u_{it}
u_i_amp	u_{iamp}	F	All	x_i displacement amplitude	$\lvert u_i \rvert$
u_i_ph	u_{iph}	F	All	x_i displacement phase	$\frac{180}{\pi}\mathrm{mod}(\mathrm{angle}(u_i), 2\pi)$
u_i_t	u_{it}	F	All	x_i velocity	$j\omega u_i$
u_i_t_amp	u_{itamp}	F	All	x_i velocity amplitude	ωu_{iamp}
u_i_t_ph	u_{itph}	F	All	x_i velocity phase	$\mathrm{mod}(u_{iph} + 90, 360)$
u_i_tt	u_{itt}	F	All	x_i acceleration	$-\omega^2 u_i$
u_i_tt_amp	u_{ittamp}	F	All	x_i acceleration amplitude	$\omega^2 u_i$
u_i_tt_ph	u_{ittph}	F	All	x_i acceleration phase	$\mathrm{mod}(u_{iph} + 180, 360)$
disp	$disp$	All	All	Total displacement	$\sqrt{\sum_i (\mathrm{real}(u_i))^2}$

TABLE 7-1: IN-PLANE TRUSS APPLICATION MODE VARIABLES

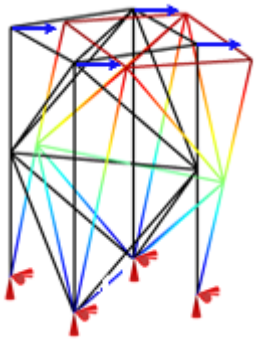
NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
xn	x_n	All	B	Parameter along edge only used for linear constraint	$\frac{x(x_2-x_1)+y(y_2-y_1)}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}}$
exixjT	ε_{xixjT}	All	B	Tangential strain tensor	If large deformation $\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}\Big\|_T+\frac{\partial u_j}{\partial x_i}\Big\|_T+\frac{\partial u_k}{\partial x_i}\Big\|_T\cdot\frac{\partial u_k}{\partial x_j}\Big\|_T\right)$ Else $\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}\Big\|_T+\frac{\partial u_j}{\partial x_i}\Big\|_T\right)$
en	ε_n	All	B	Axial strain	$t_x(\varepsilon_{xT}t_x+\varepsilon_{xyT}t_y)+t_y(\varepsilon_{xyT}t_x+\varepsilon_{yT}t_y)$
sn	σ_n	All	B	Axial stress	$E(\varepsilon_n-\alpha(T-T_{ref})-\varepsilon_{ni})+\sigma_{ni}$
exixjT_t	ε_{xixjTt}	T	B	Tangential strain rate tensor	If large deformation $\frac{1}{2}\left(\frac{\partial ut_i}{\partial x_j}\Big\|_T+\frac{\partial ut_j}{\partial x_i}\Big\|_T+\frac{\partial ut_k}{\partial x_i}\Big\|_T\cdot\frac{\partial u_k}{\partial x_j}\Big\|_T+\frac{\partial u_k}{\partial x_i}\Big\|_T\cdot\frac{\partial ut_k}{\partial x_j}\Big\|_T\right)$ Else $\frac{1}{2}\left(\frac{\partial ut_i}{\partial x_j}\Big\|_T+\frac{\partial ut_j}{\partial x_i}\Big\|_T\right)$
exixjT_t	ε_{xixjTt}	F	B	Tangential strain rate tensor	$\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}\Big\|_T+\frac{\partial u_j}{\partial x_i}\Big\|_T\right)j\omega$
exixjT_b	ε_{xixjTb}	Buckling	B	Tangential strain buckling tensor	$\frac{1}{2}\left(\frac{\partial ut_k}{\partial x_i}\Big\|_T\cdot\frac{\partial u_k}{\partial x_j}\Big\|_T+\frac{\partial u_k}{\partial x_i}\Big\|_T\cdot\frac{\partial ut_k}{\partial x_j}\Big\|_T\right)$
en_t	ε_{nt}	F/T	B	Axial strain rate	$t_x(\varepsilon_{xTt}t_x+\varepsilon_{xyTt}t_y)+t_y(\varepsilon_{xyTt}t_x+\varepsilon_{yTt}t_y)$
en_b	ε_{nb}	F/T	B	Axial buckling strain	$t_x(\varepsilon_{xTb}t_x+\varepsilon_{xyTb}t_y)+t_y(\varepsilon_{xyTb}t_x+\varepsilon_{yTb}t_y)$

TABLE 7-1: IN-PLANE TRUSS APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
sn_t	σ_{nt}	F/T	B	Axial stress rate	$E\varepsilon_{nt}$
N	N	All	B	Axial force	$A\sigma_n$
Fig	F_{ig}	S/T/E	B/P	Edge, point load in global x_i-dir.	If global coordinate system $\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \end{bmatrix}$ If other coordinate system $\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = T_{coord} \begin{bmatrix} F_x \\ F_y \end{bmatrix}$
Fig	F_{ig}	F	B/P	Edge, point load in global x_i-dir.	If global coordinate system $F_{ig} = F_i F_{iAmp} e^{jF_{iPh}\frac{\pi}{180}}$ If other coordinate system $\begin{bmatrix} F_{xg} \\ F_{yg} \end{bmatrix} = T_{coord} \begin{bmatrix} F_x F_{xAmp} e^{jF_{xiPh}\frac{\pi}{180}} \\ F_y F_{yAmp} e^{jF_{yPh}\frac{\pi}{180}} \end{bmatrix}$
Ws	W_s	All	B	Strain energy density	$\frac{A}{2}(\varepsilon_n \sigma_n)$

3D Truss Application Mode

This section describes how to use the 3D Truss application mode in the Structural Mechanics Module to model three-dimensional trusses or sagging cable-like structures.



3D Truss application modes are defined on edges in 3D. All settings for the application mode appear in "Application Mode Description" on page 180.

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, respectively.

Variables

A large number of variables are available for use in expressions and for postprocessing purposes. In addition to the variables listed below almost all application mode parameters are available as variables. Some variables are different for different analyses, which you can see in the Analysis column. For frequency response analysis, a number of new variables are available as seen in the Table 7-2. In addition to variables such as stress and strain there are also variables for the amplitude and phase of that variable that

you can access using the convention of adding _amp or _ph after the name of the variable, for example:

- en_amp is the amplitude of the bending moment in the local y direction.
- sn_ph is the phase of the axial stress.

Table 7-2 uses an index convention, where a single index on u_i means that i runs over the global displacements (u, v, and w). The Analysis column uses the following abbreviations:

ANALYSIS	ABBREVIATION
Static	S
Frequency response	F
Time dependent	T
Eigenfrequency	E

TABLE 7-2: 3D TRUSS APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
u_i	u_i	All	All	x_i displacement	u_i
u_it	u_{it}	T	All	x_i velocity	u_{it}
u_i_amp	u_{iamp}	F	All	x_i displacement amplitude	$\lvert u_i \rvert$
u_i_ph	u_{iph}	F	All	x_i displacement phase	$\frac{180}{\pi}\mathrm{mod}(\mathrm{angle}(u_i), 2\pi)$
u_i_t	u_{it}	F	All	x_i velocity	$j\omega u_i$
u_i_t_amp	u_{itamp}	F	All	x_i velocity amplitude	ωu_{iamp}
u_i_t_ph	u_{itph}	F	All	x_i velocity phase	$\mathrm{mod}(u_{iph} + 90, 360)$
u_i_tt	u_{itt}	F	All	x_i acceleration	$-\omega^2 u_i$
u_i_tt_amp	u_{ittamp}	F	All	x_i acceleration amplitude	$\omega^2 u_i$

TABLE 7-2: 3D TRUSS APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
u$_i$_tt_ph	u_{ittph}	F	All	x_i acceleration phase	$\mathrm{mod}(u_{iph}+180,360)$
disp	$disp$	All	All	Total displacement	$\sqrt{\sum_i (\mathrm{real}(u_i))^2}$
xn	x_n	All	E	Parameter along edge only used for linear constraint	$\frac{x(x_2-x_1)+y(y_2-y_1)+z(z_2-z_1)}{\sqrt{(x_2-x_1)^2+(y_2-y_1)^2+(z_2-z_1)^2}}$
exixjT	ε_{xixjT}	All	E	Tangential strain tensor	If large deformation $\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}\Big\|_T+\frac{\partial u_j}{\partial x_i}\Big\|_T+\frac{\partial u_k}{\partial x_i}\Big\|_T\cdot\frac{\partial u_k}{\partial x_j}\Big\|_T\right)$ Else $\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}\Big\|_T+\frac{\partial u_j}{\partial x_i}\Big\|_T\right)$
en	ε_n	All	E	Axial strain	$t_x(\varepsilon_{xT}t_x+\varepsilon_{xyT}t_y+\varepsilon_{xzT}t_z)+$ $t_y(\varepsilon_{xyT}t_x+\varepsilon_{yT}t_y+\varepsilon_{yzT}t_z)+$ $t_z(\varepsilon_{xzT}t_x+\varepsilon_{yzT}t_y+\varepsilon_{zT}t_z)$
sn	σ_n	All	E	Axial stress	$E(\varepsilon_n-\alpha(T-T_{ref})-\varepsilon_{ni})+\sigma_{ni}$
exixjT_t	ε_{xixjTt}	T	E	Tangential strain rate tensor	If large deformation $\frac{1}{2}\left(\frac{\partial ut_i}{\partial x_j}\Big\|_T+\frac{\partial ut_j}{\partial x_i}\Big\|_T+\frac{\partial ut_k}{\partial x_i}\Big\|_T\cdot\frac{\partial u_k}{\partial x_j}\Big\|_T+\frac{\partial u_k}{\partial x_i}\Big\|_T\cdot\frac{\partial ut_k}{\partial x_j}\Big\|_T\right)$ Else $\frac{1}{2}\left(\frac{\partial ut_i}{\partial x_j}\Big\|_T+\frac{\partial ut_j}{\partial x_i}\Big\|_T\right)$
exixjT_t	ε_{xixjTt}	F	E	Tangential strain rate tensor	$\frac{1}{2}\left(\frac{\partial u_i}{\partial x_j}\Big\|_T+\frac{\partial u_j}{\partial x_i}\Big\|_T\right)j\omega$

TABLE 7-2: 3D TRUSS APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
exixjT_b	ε_{xixjTb}	Buckling	E	Tangential strain buckling tensor	$\frac{1}{2}\left(\frac{\partial ut_k}{\partial x_i}\bigg\|_T \cdot \frac{\partial u_k}{\partial x_j}\bigg\|_T + \frac{\partial u_k}{\partial x_i}\bigg\|_T \cdot \frac{\partial ut_k}{\partial x_j}\bigg\|_T\right)$
en_t	ε_{nt}	F/T	E	Axial strain rate	$t_x(\varepsilon_{xTt}t_x + \varepsilon_{xyTt}t_y + \varepsilon_{xzTt}t_z) +$ $t_y(\varepsilon_{xyTt}t_x + \varepsilon_{yTt}t_y + \varepsilon_{yzTt}t_z) +$ $t_z(\varepsilon_{xzTt}t_x + \varepsilon_{yzTt}t_y + \varepsilon_{zTt}t_z)$
en_b	ε_{nb}	F/T	E	Axial buckling strain	$t_x(\varepsilon_{xTb}t_x + \varepsilon_{xyTb}t_y + \varepsilon_{xzTb}t_z) +$ $t_y(\varepsilon_{xyTb}t_x + \varepsilon_{yT}t_y + \varepsilon_{yzTb}t_z) +$ $t_z(\varepsilon_{xzTb}t_x + \varepsilon_{yzTb}t_y + \varepsilon_{zTb}t_z)$
sn_t	σ_{nt}	F/T	E	Axial stress rate	$E\varepsilon_{nt}$
N	N	All	E	Axial force	$A\sigma_n$
Fig	F_{ig}	S/T/E	E/P	Edge, point load in global x_i-dir.	If global coordinate system $\begin{bmatrix} F_{xg} \\ F_{xg} \\ F_{zg} \end{bmatrix} = \begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix}$ If other coordinate system $\begin{bmatrix} F_{xg} \\ F_{yg} \\ F_{zg} \end{bmatrix} = T_{coord}\begin{bmatrix} F_x \\ F_y \\ F_y \end{bmatrix}$

TABLE 7-2: 3D TRUSS APPLICATION MODE VARIABLES

NAME	SYMBOL	ANALYSIS	DOMAIN	DESCRIPTION	EXPRESSION
Fig	F_{ig}	F	E/P	Edge, point load in global x_i-dir.	If global coordinate system $F_{ig} = F_i F_{iAmp} e^{jF_{iPh}\frac{\pi}{180}}$ If other coordinate system $\begin{bmatrix} F_{xg} \\ F_{yg} \\ F_{zg} \end{bmatrix} = T_{coord} \begin{bmatrix} F_x F_{xAmp} e^{jF_{xiPh}\frac{\pi}{180}} \\ F_y F_{yAmp} e^{jF_{yPh}\frac{\pi}{180}} \\ F_z F_{zAmp} e^{jF_{zPh}\frac{\pi}{180}} \end{bmatrix}$
Ws	W_s	All	E	Strain energy density	$\frac{A}{2}(\varepsilon_n \sigma_n)$

8

User-Defined Classes in COMSOL Script 1.0b

Using COMSOL Script it is possible to define new data types, classes, and to create objects that are instances of these classes. A class is an aggregate of fields and methods.

The class model used is inspired by Java's class system.

This chapter describes the use of classes and objects in COMSOL Script 1.0b.

Introductory Example: Rectangle

The file `Rectangle.csl` defines the class `Rectangle`:

```
class Rectangle
%A class for rectangles.

public x0 x1  % x0<x1
public y0 y1  % y0<y1

function Rectangle(varargin)
%RECTANGLE(X0, Y0, X1, Y1) creates a rectangle with vertices
%in (x0,y0) and (x1,y1).
switch nargin
case 4
  in = varargin;
  [x0 x1] = deal(min(in{1}, in{3}), max(in{1}, in{3}));
  [y0 y1] = deal(min(in{2}, in{4}), max(in{2}, in{4}));
case 1
  r = varargin{1};
  [x0 x1] = [r.x0 r.x1];
  [y0 y1] = [r.y0 r.y1];
otherwise
  error('Usage: Rectangle(x0, y0, x1, y1)');
end

public function out = area
%AREA returns the area of the rectangle.
out = (x1-x0)*(y1-y0);

public static function out = overlap(r1, r2)
%OVERLAP(R1, R2) returns the area of the overlap between the
%two rectangles R1 and R2.
out = max(min(r1.x1, r2.x1)-max(r1.x0, r2.x0), 0)*...
      max(min(r1.y1, r2.y1)-max(r1.y0, r2.y0), 0);
```

This defines a Rectangle *class* with the four *fields* x0, y0, x1, and y1, a *constructor*, and the two *methods* `area` and `overlap`. To create a `Rectangle` *object*, use the same syntax as for a function call:

```
r = Rectangle(1, 3, 2, 9)
r =
Rectangle object
  x0:  [1]
  x1:  [2]
  y0:  [3]
  y1:  [9]
```

```
r.area

ans =
6
```

The call `Rectangle(1, 3, 2, 9)` returns a rectangle with vertices in (1,3) and (2,9). The variable `r` is of the type `Rectangle`; it is an *instance* of the `Rectangle` class. There can only be one definition of a class, but there can be many instances of it. By default, the fields of an object are displayed in the same way as if it were a structure.

The call `r.area` invokes the `area` method of the object r. The syntax for invoking a method without arguments is the same as for accessing a field of a structure.

The class contains three methods: the constructor `Rectangle`, the accessor `area`, and the static function `overlap`. The method declarations look like declarations of local functions in a function M-file, but unlike local functions, they can be accessed outside the class.

`overlap` is an example of a static method: it is invoked when `overlap` is invoked with arguments where at least one is a `Rectangle`:

```
r = Rectangle(1, 5, 6, 8);
s = Rectangle(3, 2, 5, 7);
overlap(r, s)

ans =
4
```

The overlap method is not visible if none of the arguments is a `Rectangle`: for example, `overlap(1,2)` gives an error.

The Structure of the Class File

A class is defined by a `.csl`-file on the path that has the following contents:

- Class header
- Precedence declarations
- Field declarations
- Method declarations

Only the class header is mandatory. Below you find brief descriptions of the contents of each part of the file. The next section contains a more detailed description.

Class Header

The class header declares the name of the class, its copy semantics, and its superclass, if any. The most basic form is a value class with no superclass:

```
class Rectangle
```

The name of the class, here `Rectangle`, must coincide with the filename, `here Rectangle.csl`. That the class is a value class means that a copy is made whenever an instance of the class is used as argument to a function, or is used in an assignment. All other data types except for Java objects have value semantics. The opposite is a reference class, declared with the `reference` modifier:

```
reference class Rectangle
```

Reference semantics means that a true copy of the object is never made, only references to the same object. This is the semantics that Java objects have.

The superclass of a class is declared using the `extends` keyword:

```
class Rectangle extends Shape
```

The superclass must be defined by a `.csl`-file on the path.

Precedence Declarations

This optional section contains the classes that have higher or lower precedence than the class being declared. It contains zero or more lines of the form

```
superiorto <classname>
```

or

```
inferiorto <classname>
```

Field Declarations

This optional section contains zero or more field declarations of the forms

```
<access modifiers> <name> = <expression
```

or

```
<access modifiers> <name1> [<name2> ...]
```

The fields declared are like the fields of a structure, with the difference that a class always contains the same fields. The access modifiers control where the field is visible. You must specify one of `public`, `protected`, and `private`, and you can optionally use `static` and `transient`.

The first syntax allows for initial values to be supplied:

```
public version = 1;
```

This declares the field `version` with the default value 1. This means that the `version` field is assigned to 1 when an instance of the class is created. Fields without initial values are assigned to [].

Method Declarations

This optional section contains zero or more method declarations of the form

```
[<access modifiers>] <function declaration>
```

You can specify one of the access modifiers `public`, `protected`, and `private`, as well as `static`. The method is considered `public` unless you specify `protected` or `private`.

Except for the optional access modifiers, the syntax of a function declaration is the same as for a local function declaration in an M-file.

Access Modifiers

You can assign access modifiers to the fields and methods of a class. There are three types of modifiers:

- Visibility modifiers: `public`, `protected`, and `private`. They specify where the member is visible: `public` means that the member is visible everywhere, `protected` that it is visible in the class and classes inheriting from it, and `private` that it is visible only in the class where it was declared.
- Static modifier: `static`. A static member is one that is associated with the class, not with an instance of the class. The opposite is an instance member (the default).
- Transient modifier: `transient`. COMSOL Script does not save a field marked as transient when you run the command `save`. You can use this to avoid saving temporary data that is easy to recompute.

Each member field must have a visibility modifier specified. For methods this is not necessary; if omitted, the default visibility is `public`.

A class where all fields are `public` behaves a lot like a structure: It is possible to read and assign values to all fields using the `var.field` syntax. Restricting the visibility can be useful for hiding class internals that are inconsequential for the user of the class.

Member Fields

A class can be viewed as a structure with a predefined set of fields, the instance (non-static) fields. Static fields have semantics similar to `global` or `persistent` variables.

Instance Fields

Fields not declared as static are instance fields: They belong to an instance of the class, like fields in a structure. The fields of the `Rectangle` class are examples of instance fields:

```
class Rectangle

public x0 x1  % x0<x1
public y0 y1  % y0<y1
```

Outside of the class, you can use these fields like the fields of a structure:

```
r = Rectangle(2, 3, 6, 7);
r.x1

ans =
6
```

This is only possible for `public` fields; you cannot access `protected` and `private` fields this way. It is also possible to modify public fields like fields of a structure:

```
r = Rectangle(2, 3, 6, 7);
r.y1 = 10

r =
Rectangle object
  x0:  [2]
  x1:  [6]
  y0:  [3]
  y1:  [10]
```

When a non-static method of a class is run, you can access its instance fields like local variables:

```
public function out = area
%AREA returns the area of the rectangle.
out = (x1-x0)*(y1-y0);
```

The values of `x0`, `x1`, `y0`, and `y1` are taken from the instance fields, and any assignments to them would result in the instance fields being changed.

Static Fields

Fields declared `static` belong to the class, not any instance of it. As an example, consider the class `MathConst`:

```
class MathConst

% The golden ratio
public static golden = (sqrt(5)+1)/2;
% Perfect numbers less than 10^11
public static perfect = [6 28 496 8128 33550336 8589869056];
```

The two fields `golden` and `perfect` are static. Their values are set using initializers, (see the next section for an explanation of this). To access them, access the class like a structure:

```
MathConst.golden

ans =
1.6180

MathConst.perfect(2:3)

ans =
28  496
```

It is also possible to change their values:

```
MathConst.golden = -17
```

Initialization of Fields

The `MathConst` class illustrates initialization:

```
public static golden = (sqrt(5)+1)/2;
```

The above means that when the class is loaded, the static field `golden` is assigned the expression in the right-hand side. For instance fields, initializers provide initial values when the object is created. As an example, consider the point class:

```
class Point

public x = 2;
public y = 5;
```

The default placement of the point is at coordinates (2, 5):

```
p = Point

p =
Point object
  x:  [2]
  y:  [5]
```

If a field is not given an initializer, it is assigned to [ ]. More specifically: When an object is created, all instance fields are initialized to [ ]. Then initializers, if any, are evaluated in the order the fields are declared. This means that

```
public x;
public y = {x 5};
public z = length(y);
```

is equivalent to

```
public x = [];
public y = {[] 5};
public z = 2;
```

Member Methods

Constructor

The constructor is a function with the same name as the class and is called when an instance of the class is created. The purpose of the constructor is to set up a valid object, possibly using input arguments. It cannot be static and must not return anything. As an example, consider the constructor of the `Rectangle` class:

```
function Rectangle(varargin)
%RECTANGLE(X0, Y0, X1, X1) creates a rectangle with vertices
%in (x0,y0) and (x1,y1).
switch nargin
case 4
  in = varargin;
  [x0 x1] = deal(min(in{1}, in{3}), max(in{1}, in{3}));
  [y0 y1] = deal(min(in{2}, in{4}), max(in{2}, in{4}));
...
```

The constructor is called whenever a Rectangle is created: When

```
r = Rectangle(1, 3, 2, 9)
```

is executed, an empty `Rectangle` object is created. At this point, the four instance fields `x0`, `x1`, `y0`, and `y1` have been initialized with [ ]. Then the constructor is invoked for the new object. The constructor is an instance method and can therefore modify the instance fields: The assignments to, for example, `x0` modify the field `x0` of the instance. The methods of a class differ from local functions in this respect: A local function `Rectangle` with the above contents would assign values to `x0` in its own workspace, but when leaving the function, these values would be lost.

A class does not need a constructor: If there is no constructor, the fields are assigned default values if provided, otherwise [ ].

Instance Methods

An instance method is a method that is not declared static. It operates on an instance of a class and has access to the instance fields. The `area` method in the `Rectangle` class is an example of an instance method:

```
public function out = area
%AREA returns the area of the rectangle.
out = (x1-x0)*(y1-y0);
```

An instance method can read or write the instance fields like local variables, just like the constructor. `area` uses the coordinates of the rectangle to compute its area.

Static Methods

A static method has access to the static fields of the class but not to any instance fields as it cannot be run for any instance of the class. The `overlap` method of the `Rectangle` class is static:

```
public static function out = overlap(r1, r2)
%OVERLAP(R1, R2) returns the area of the overlap between the
%two rectangles R1 and R2.
out = max(min(r1.x1, r2.x1)-max(r1.x0, r2.x0), 0)*...
      max(min(r1.y1, r2.y1)-max(r1.y0, r2.y0), 0);
```

There are two way to invoke a static method:

- Calling it like a function with one or more objects as arguments:

```
r = Rectangle(1, 5, 6, 8);
s = Rectangle(3, 2, 5, 7);
overlap(r, s)

ans =
4
```

- Specifying it explicitly:

```
Rectangle.overlap(r, s)

ans =
4
```

The first syntax works because at least one of the arguments to `overlap` is an object, here of the class `Rectangle`, and the class `Rectangle` has a visible static method called `overlap`.

Inheritance

Sometimes a class is a specialization or extension of another class. You can specify this relation in the class header using the `extends` keyword:

```
class Rectangle extends Shape
```

A rectangle is a specialization of the shape concept, and it therefore makes sense to let the class `Rectangle` inherit from the class `Shape`, which then becomes the *superclass* of `Rectangle`, the *derived class*. The members and methods of the superclass are visible in the derived class. Suppose that the Shape and Rectangle classes contain the following fields:

```
class Shape
public name
...

class Rectangle extends Shape
public x0 x1
public y0 y1
...
```

The `Rectangle` class contains five fields: The four fields declared in `Rectangle`, and the field declared in the superclass, `Shape`. For a user of the class, there is no distinction between the two types of fields:

```
r = Rectangle(1, 3, 2, 9);
r.name = 'COMSOL';
r.name

ans =
  COMSOL
```

Reference and Value Classes

Differences

When you make an assignment `a = b`, COMSOL Script normally assigns a copy of `b` to `a`. This is the case for all data types except for Java objects and instances of reference classes: For these data types, only a reference is copied. Consider the class `RefClass` defined as follows:

```
reference class RefClass
public x = 5;
```

The software only copies a reference when an assignment is made:

```
r = RefClass;
s = r;
r.x = 10;
s.x

ans =
10
```

Consider on the other hand the class `ValueClass`, defined as follows:

```
class ValueClass
public x = 5;
```

For a value class, the assignment makes a true copy:

```
v = ValueClass;
s = v;
v.x = 10;
s.x

ans =
5
```

The same rules apply when passing arguments to functions: For an instance of a value class, a copy of the value is passed; for an instance of a reference class, a reference to the value is passed.

Choosing Class Type

The main factor determining the class type is how you view the class:

- If you think of the class as a structure augmented with methods, declare it as a value class.
- If you think of the class as a lightweight Java class, declare it as a reference class.

A reference class cannot inherit from a value class, or vice versa. Thus the choice of class type for a base class affects the entire class hierarchy.

If you define several different classes, try to use the same class type for all of them. Mixing class types can easily lead to bugs as the difference in semantics is subtle.

Built-In Object Functions

Functions That You Can Only Use for Objects

THE THIS FUNCTION

Inside an instance method, `this` returns the instance for which the method is run.

THE SUPER FUNCTION

When a constructor is run in a derived class, you can use `super` to run the constructor of the superclass. Suppose `Rectangle` inherits from `Shape`:

```
class Rectangle extends Shape

...

function Rectangle(varargin)
super(varargin);
...
```

The call `super(varargin)` runs the constructor of the `Shape` class.

It is also possible to use `super` as a structure to access or modify visible fields or methods from the superclass: `super.a = 17;` sets the field `a` in the superclass to 17. This can be useful if a member in the superclass has been shadowed by a member with the same name in the derived class.

THE CLONE FUNCTION

The `clone` function creates a true copy of an object. For a value object, this has no effect, but for a reference object it is necessary in order to copy the contents, not only a reference to the object. The class `RefClass` was above defined as

```
reference class RefClass
public x = 5;
```

Use the `clone` function to create a true copy of a `RefClass` object:

```
r = RefClass;
s = clone(r);
r.x = 10;
s.x

ans =
5
```

Without `clone`, only a reference would be copied.

Functions with Special Semantics for Objects

CLEAR CLASSES

`clear classes` removes all variables, just like `clear` does, but it also tries to remove all class definitions. This is necessary if the definition of a class changes: Unless you perform `clear classes`, COMSOL Script uses the old class definition even if the class file on disk changes.

`clear classes` can only remove the definitions of classes of which there are no instances. This means that sometimes not all classes are removed: If `clear classes` is called from a function and there still are instances of some class in the root workspace, then that class cannot be cleared. The same thing can happen if you store objects in global or persistent variables.

HELP FOR A CLASS

`help` for a class displays the comment block following the class header as well as the comment blocks following each public nonstatic method:

```
help Rectangle

A class for rectangles.

RECTANGLE(X0, Y0, X1, Y1) creates a rectangle with vertices
in (x0,y0) and (x1,y1).

AREA returns the area of the rectangle.

OVERLAP(R1, R2) returns the area of the overlap between the
two rectangles R1 and R2.
```

You can also retrieve the help text for a method:

```
help Rectangle.overlap

OVERLAP(R1, R2) returns the area of the overlap between the
two rectangles R1 and R2.
```

THE FIELDNAMES FUNCTION

For an object, `fieldnames` returns the names of the instance fields that are visible from the workspace where `fieldnames` is called:

```
r = Rectangle(2, 3, 6, 7);
```

```
fieldnames(r)

ans =
  'x0'
  'x1'
  'y0'
  'y1'
```

All the fields of the `Rectangle` class are `public`, and `fieldnames` therefore returns them. `private` and `protected` members are only returned by `fieldnames` when invoked from a method of the class.

THE METHODS FUNCTION

`methods` displays the methods declared by a class:

```
methods('Rectangle')

class Rectangle
  public function Rectangle
  public function area
```

By default, `methods` only displays public nonstatic methods. It is possible to select methods to display based on access modifiers:

```
methods('Rectangle', 'static')

class Rectangle
  public static function overlap
```

The second argument to `methods` is a string or cell array of strings that lists all access modifiers to include.

When requesting output, `methods` returns a cell array:

```
c = methods('Rectangle', 'static')

c =
{'overlap'}
```

THE STRUCT FUNCTION

When invoked for an object, `struct` returns a structure containing the fields of the object that are visible in the workspace where `struct` is called:

```
r = Rectangle(1, 3, 2, 9);
struct(r)

ans =
  x0:  [1]
  x1:  [2]
```

```
y0:  [3]
y1:  [9]
```

Overloading

Overloading Operators

You can overload all unary and binary arithmetic, relational, and logical operators, as well as the concatenation operators [,] and [;]. To overload an operator, create a `public static` function that defines the semantics of the overloaded function. As an example, consider a class `Rational` that represents rational numbers:

```
class Rational
%RATIONAL is a class for exact representation of a rational number
%as a ratio between integers.

private a  % Numerator
private b  % Denominator, always > 0

...

static function out = plus(r1, r2)
%PLUS Sum.
%   OUT = PLUS(R1, R2) returns the sum of R1 and R2.
r1 = Rational(r1);
r2 = Rational(r2);
out = Rational(r1.a*r2.b+r1.b*r2.a, r1.b*r2.b);

...
```

The static function `plus` provides an overload for the + operator:

```
Rational(1,3)+Rational(1/4)  % 1/3+1/4 = 7/12
  7 / 12
Rational(pi)+1               % uses 355/113 as approximation of pi
  468/113
```

A demonstration example in this release includes the complete definition of the `Rational` class. Run `help Rational` or `type Rational` to see its contents.

Each operator has a corresponding function that the software invokes when at least one of the operands is an object. This is the function that the class must provide for it to define an overloaded operator. The example above includes overloading of the +

operator by defining the `plus` member method. Table 8-1 contains the complete map between operators and functions:

TABLE 8-1: OPERATORS AND THEIR CORRESPONDING FUNCTIONS

OPERATOR	FUNCTION
+ (unary, +a)	uplus
+ (binary, a+b)	plus
- (unary, -a)	uminus
- (binary, a-b)	minus
*	mtimes
.*	times
/	mrdivide
./	rdivide
\	mldivide
.\	ldivide
^	mpower
.^	power
==	eq
~=	ne
>=	ge
>	gt
<=	lt
<	le
&	and
|	or
~	not
[, ,]	horzcat
[; ;]	vertcat
:	colon
'	ctranspose
.'	transpose

FIELD READ/WRITE

If a class has an instance method called `fieldread`, then COMSOL Script calls that function when it reads a field of an object using the `var.field` syntax. The `fieldread` method must take one input argument and return one output:

```
function out = fieldread(field)
```

The `field` argument is the string that follows the `.` (dot) in `var.field`. By overloading `fieldread` it is possible to let a class behave as if it has fields that it does not have. You can use this overloading for making the representation of the data independent of the interface to the class.

Similarly, if a class has an instance function called `fieldwrite`, then that function is called when a field of an object is written using the `var.field = val` syntax. It must take two input arguments and not return anything:

```
function fieldwrite(field, val)
```

The `field` parameter is the string that followed the `.` in `var.field` and the `val` parameter is the value to which the field was assigned. You can use `fieldwrite` when the fields of the class have dependencies: Changing the value of one field can force the recalculation of others.

ARRAY READ/WRITE

An object can only be indexed using parentheses if it has an instance method called `arrayread`: If `c` is an object, then `c(args)` is interpreted as `c.arrayread(args)`, where `arrayread` is invoked for the arguments (one or more) and is expected to return one value:

```
function out = arrayread(args)
```

Overloading `arrayread` can be useful for classes where the parentheses denote evaluation, for instance in a class representing a mathematical function of one or more variables.

Similarly, array assignment using the syntax `c(args) = val` is interpreted as `c.arraywrite(args, val)`.

CELL ARRAY READ/WRITE

It is only possible to index an object using curly brackets if it has an instance method called `cellread`: If `c` is an object, then `c{args}` is interpreted as `c.cellread(args)`

where `cellread` is invoked for the arguments (one or more) and is expected to return one value:

```
function out = cellread(args)
```

Similarly, cell array assignment using the syntax `c{args} = val` is interpreted as `c.cellwrite(args, val)`.

Overloading Save and Load

When an object is saved to file using `save`, the default behavior is to save all nontransient instance fields. It is possible to override this behavior by a class providing a `writeobject` method. This method must have the interface

```
public function out = writeobject
```

COMSOL Script writes the output from `writeobject` to file when the object is saved.

If the class has an overloaded `writeobject` method it usually also has to provide an overloaded `readobject` method: This method is called when a object is loaded from file using `load`. It must have the interface

```
public function readobject(data)
```

When the software loads an object, it first creates an empty instance of its class. Then the `readobject` method is invoked, and the data argument is the output from `writeobject` when the object was saved.

You can overload `readobject` but not `writeobject`. In this case, the input to `readobject` is the default representation of the object: As a cell array with one column for each level of the class hierarchy. Consider the `Rectangle` class inheriting from `Shape`:

```
class Shape
public name
...

class Rectangle extends Shape
public x0 x1
public y0 y1
```

A `Rectangle` object with vertices in $(2, 3)$ and $(5, 7)$ and the name `'MyRect'` would be saved as the cell array

```
{'Shape'                'Rectangle'
 struct('name','MyRect') struct('x0',2,'x1',5,'y0',3, y1',7}
```

This would be the argument to an overloaded `readobject` in `Rectangle` if no overloaded `writeobject` was present. There is one column for each level of the class hierarchy. The first row contains the name class names, and the second row contains a structure with one field for each non-transient field on that level of the hierarchy.

Overloading Display

The default way to display an object is to display its public fields like the fields of a structure:

```
r = Rectangle(1, 3, 2, 9)

r =
Rectangle object
  x0:  [1]
  x1:  [2]
  y0:  [3]
  y1:  [9]
```

If the class has a public nonstatic function called `display`, COMSOL Script invokes it when it displays an object of the class. You can extend the `Rectangle` class with such a method:

```
public function display
sprintf('Rectangle with corners (%.2f,%.2f) and (%.2f,%.2f).', ...
        x0, y0, x1, y1)
```

This results in the following output:

```
r = Rectangle(1, 3, 2, 9)

ans =
Rectangle with corners (1.00, 3.00) and (2.00, 9.00).
```

Precedence

Precedence Between Functions and Methods

Methods in different classes can have the same name, and method names can also coincide with names of functions, both built-in functions and M-files. COMSOL Script resolves a function invocation of the form `func(arg1, ...)` by considering possible interpretations in the following order:

1. As an array index expression, if there is a variable called `func` in the workspace.
2. As the invocation of the static method `func` of the class of any object in the argument list.
3. As a call to the built-in function `func`.
4. As a call to the method `func` in the class where execution currently takes place.
5. As a call to the user-defined function `func`.
6. Interpretations 3–5 tried using case-insensitive name matching.

The `Rectangle` class contains a static method called `overlap`. Suppose that there also is an M-file, `overlap.m`. By rule 2 above,

```
overlap(Rectangle(1,2,3,4), Rectangle(5,6,7,8))
```

invokes the `overlap` method of the `Rectangle` class, but

```
overlap(5, 7)
```

instead calls `overlap.m`.

Precedence Between Methods from Different Classes

When a function call contains several object arguments, COMSOL Script normally considers classes in the order they appear in the argument list: Suppose that the classes `Rectangle` and `Circle` both have a static `overlap` method. Then

```
overlap(Rectangle(2, 3, 4, 5), Circle(1, 2, 3))
```

is interpreted as

```
Rectangle.overlap(Rectangle(2, 3, 4, 5), Circle(1, 2, 3))
```

The `overlap` method of the `Circle` class would only be run if there were no `overlap` method in `Rectangle`.

In the example above, the two classes `Rectangle` and `Circle` have equal precedence. You can specify the precedence between classes using `superiorto` and `inferiorto` in the class file. The classes listed in a `superiorto` declaration have lower priority than the current class, those listed in an `inferiorto` declaration have higher priority.

The `Circle` class can be modified as follows:

```
class Circle

superiorto Rectangle
```

With this change, the methods of the `Circle` class are always given priority over methods in the `Rectangle` class when the argument list contains `Circle` and `Rectangle` objects. Then

```
overlap(Rectangle(2, 3, 4, 5), Circle(1, 2, 3))
```

is interpreted as

```
Circle.overlap(Rectangle(2, 3, 4, 5), Circle(1, 2, 3))
```

It is possible to combine several `superiorto` and `inferiorto` declarations in the same class:

```
class Circle

superiorto Rectangle Square
inferiorto Hexagon
```

Note: Excessive use of `superiorto` and `inferiorto` makes the program flow hard to follow.

Using Classes as Packages

You can use classes to bundle groups of functions together: A class without instance fields where all methods are `public` and `static` can serve as a container for a set of related functions. As an example, consider the following class:

```
class MyMaths

static function y = sinc(x)
%SINC(X) returns SIN(X)/X if X is nonzero, otherwise 1.
y = ones(size(x));
ix = find(x~=0);
y(ix) = sin(x(ix))./x(ix);

static function y = smhs(x, scale)
%SMHS(x) approximates the step function Y=(X>0) by smoothing the
%transition within the interval -SCALE < X < SCALE.
x=x./scale;
y=(x>-1 & x<1).*(0.5+x.*(0.75-0.25*x.*x))+(x>=1);
```

You can view `MyMaths` as a package that contains the two functions `sinc` and `smhs`:

```
MyMaths.sinc(0:3)

ans =
         1     0.8415     0.4546     0.0470
```

By creating a class containing a group of related functions, preferably small, it becomes easier to maintain the functions, as only one file is ever changed. Sharing code between functions also becomes easier.

INDEX