

Heiki Kasemägi (Tartu Ülikool), 2013



E-kursuse

# „COMPUTATIONAL PHYSICS II“

materjalid

Aine maht 3 EAP

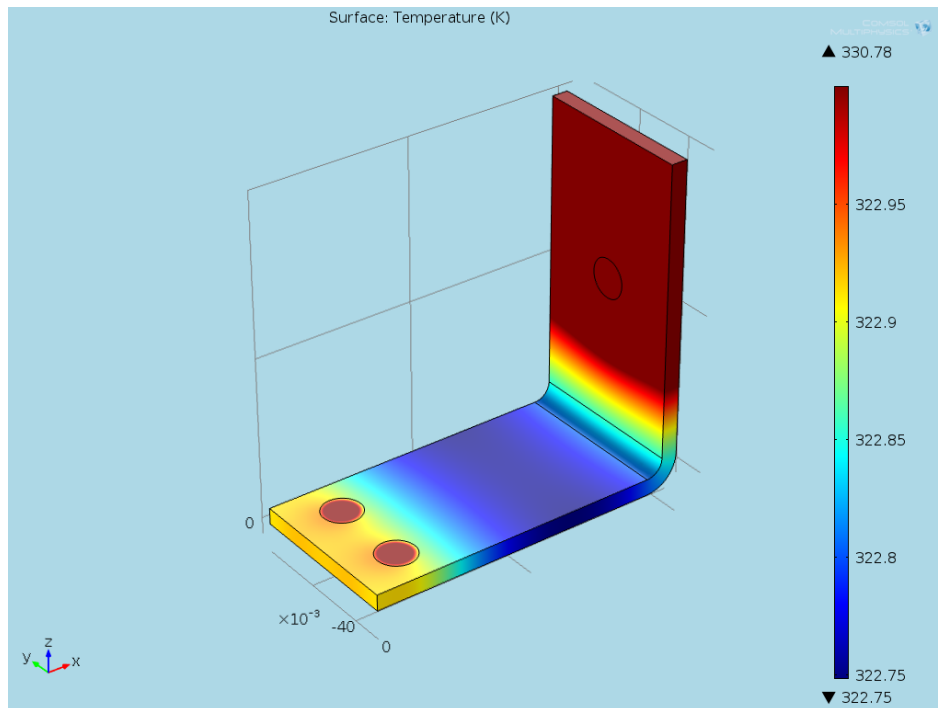
Heiki Kasemägi (Tartu Ülikool), 2013

# Computational Physics II

## Practical Work

### Comsol Multiphysics

#### Introduction



Heiki Kasemägi

October 17, 2013

# Contents

<b>Introduction</b>	<b>2</b>
Computer Simulations . . . . .	2
<b><i>COMSOL Multiphysics</i></b> . . . . .	2
Passing the Practical Work . . . . .	3
References . . . . .	3
<b>Example: the Busbar</b>	<b>4</b>
<b><i>COMSOL Multiphysics</i></b> Desktop . . . . .	4
Electrical Heating in a Busbar . . . . .	4
<i>Model Wizard</i> . . . . .	6
<i>Global Definitions</i> . . . . .	7
<i>Creating the Geometry</i> . . . . .	9
Importing the Geometry from the File . . . . .	23
Material . . . . .	25
Physics . . . . .	27
Mesh . . . . .	30
Simulation . . . . .	32
Results . . . . .	32
Generating the Images . . . . .	35

# Introduction

## Computer Simulations

The computer simulation has become a daily research method in the science and engineering. It has particularly important rôle in the development or optimization of the new products, design solutions of compounds etc. Nowadays, a wide range of simulation method are available starting from basic programming languages upto several high-level packages containing the most advanced simulation methods. Although each of them has the unique properties, the one questions or concern ties them all together:

**whether the results can be trusted?**

In order to consider the facts, which makes the software trustworthy, it is useful to keep in mind the following objective:

**goal is a model that accurately represents the reality.**

Computer simulation environment simply converts the physical laws of the real world into the virtual form. The resulting model determines the accuracy of these simplifications, the number and nature of the transformations are made.

## *COMSOL Multiphysics*

*COMSOL Multiphysics* offers a complete environment to perform computer simulations. Its graphical user interface includes the features for the CAD modelling, drawings and images import, material properties and equations to define, mesh generation, problem solving, visualization and post-processing.

*COMSOL Multiphysics* offers the opportunity to create your own model of the problem, which includes all physical phenomena that occur in the real world. Each built-in physics interface can be combined with any other physics interface. This reflects the reality taking place, for example, where an electric current is always associated with thermal phenomena.

One of the *COMSOL Multiphysics* features is an adaptability. If the model requires the introductions of a new physical phenomenon, the only thing to do is just to add it. If one needs a formula for an input, just add it. Parameterized geometry, interactive meshing and user-customized solvers are just a few examples what turn the modeling faster and smoother.

*COMSOL Multiphysics* helps to understand the problem then starting a new project. It is possible to test a variety of geometrical and physical characteristics to shiny polis the model. *COMSOL Multiphysics* offers the tools for fast and smooth construction of “what-happens-if” analysis. The optimization tools lead the model to the truly high level.

## **Passing the Practical Work**

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading “**Task #**”. The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may rise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size, title(s), legend(s) etc. The report and accompanying files are submitted via moodle.

## **References**

“Introduction to COMSOL Multiphysics” Ver. 4.3.a.

# Example: the Busbar

## *COMSOL Multiphysics* Desktop

*COMSOL Multiphysics* default desktop is divided into the three section in its default configuration (Fig. 1):

- *Model Builder* in left;
- parameters in the middle panel;
- rendering environment in right.

*Model Builder* includes the model tree, which incorporates all modeling steps and all model components starting from the global variables upto the final report. Right mouse click on the every element of the modelling tree opens an additional menu. The elements can be moved up and down in the modeling tree.

If the model tree element requires insertion of data, then the input window opens for it in the middle panel.

The *Model Builder* reflects all steps taken in their order, for example, the elements of the geometry added in the specific order to create the correct geometrical model. These steps can be edited and updated without repeating the whole simulation. The solver, what includes several solution steps, also benefits from this feature.

## Electrical Heating in a Busbar

In order to get acquainted with *COMSOL Multiphysics*, the best way is to go through an example, what contains the essential steps of the creation of the model, brings forward several *COMSOL Multiphysics* features and demonstrates the general simulation manners, and finally converges into a realistic multiphysical model.

The present model analyses a busbar (Fig. 2), what should drive the electrical current into an electrical device. The current through the busbar from screw 1 to screws 2a and 2b (Fig. 3) generates the heat due to the ohmic losses. This phenomenon is known as Joule Heating. The busbar consists of copper, the bolts are made of titan. The choice of materials is important at this point, since titan has lower electrical conductivity compared to copper resulting in higher power density.

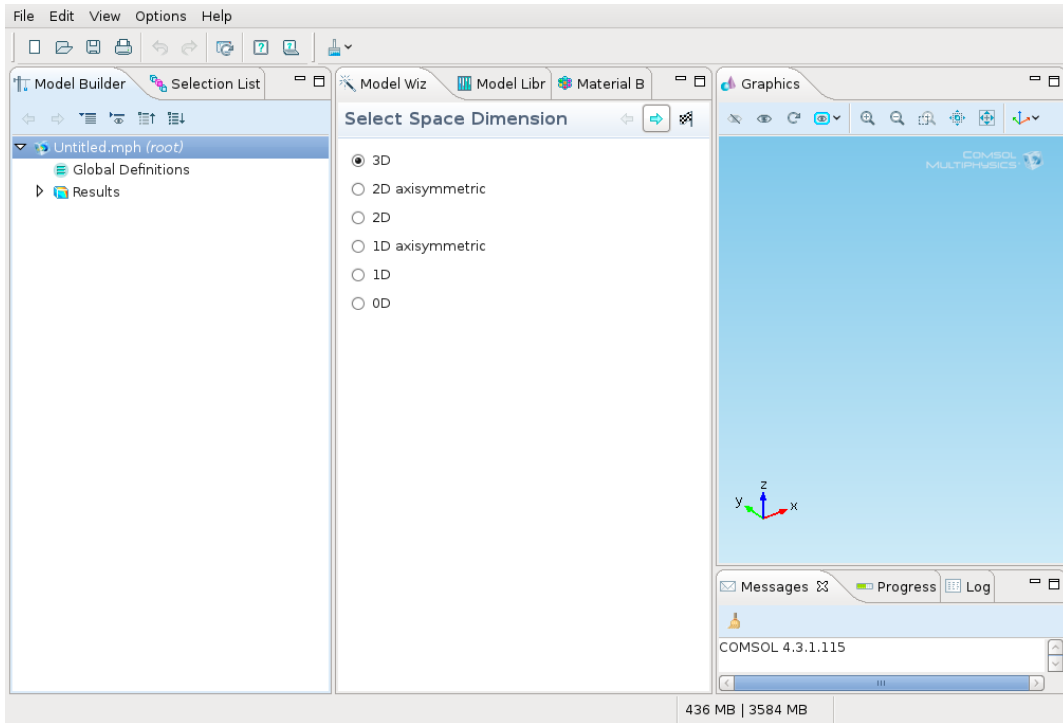


Figure 1: *COMSOL Multiphysics* default desktop.



Figure 2: The busbar.

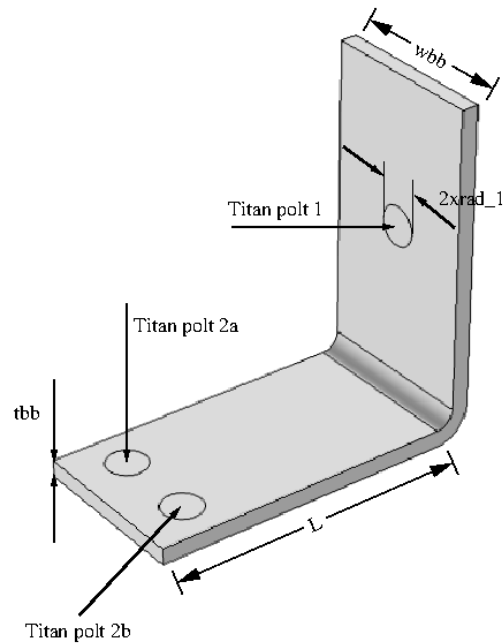


Figure 3: The constructions and dimensions of the busbar.

The aim of the simulation is to calculate precisely how much the busbar heats up. If the major physical phenomena are present, the structural stresses and deformations due to the thermal expansion, also the cooling by an air stream can be added to the model.

The Joule Heating is described by the conservation laws for the electric current and energy. The solutions of these equations are the temperature and electric field. All surfaces are cooled by the natural air convection, except the contact surfaces of the polts, which are also not heated by the external sources. The potential at the surface of the polt 1 is 20 mV and 0 V at the surfaces of the polts 2a and 2b.

## ***Model Wizard***

1 Open *Model Wizard* by starting ***COMSOL Multiphysics***.

If ***COMSOL Multiphysics*** already works, than either:

- click the ***New*** button located in the main button bar;
- select ***File*** → ***New*** from the main menu;
- click with the right mouse button on the *root* element and select ***Add Model***.



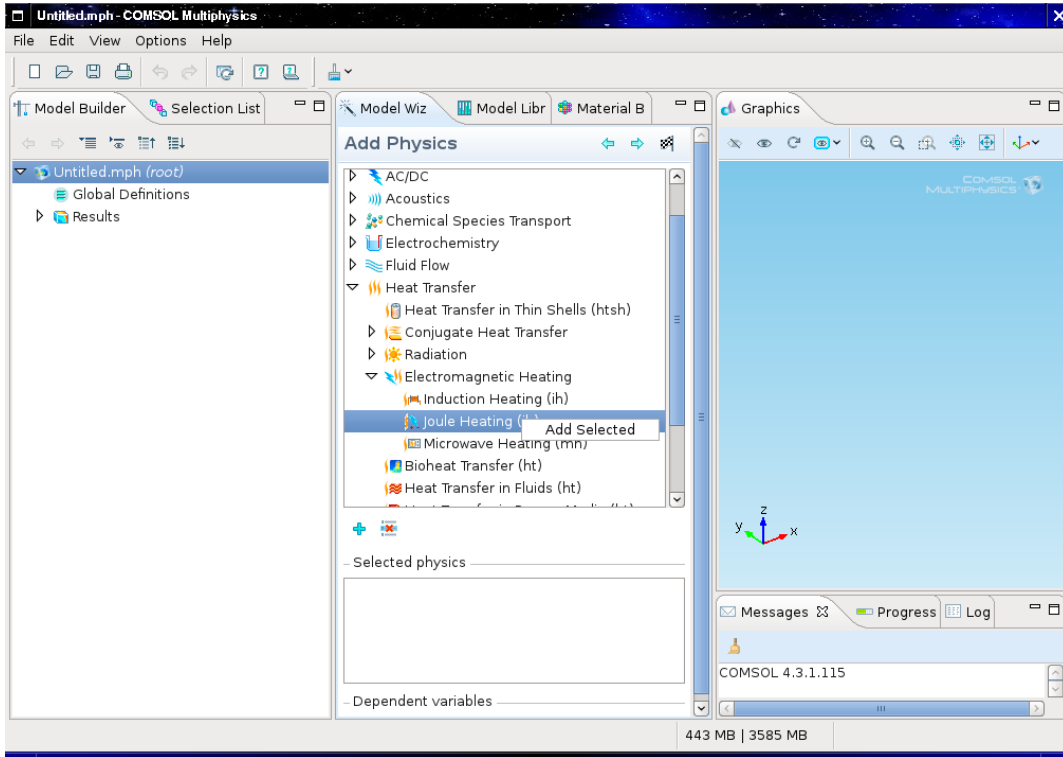


Figure 4: Adding the physics interface.

- 2 If *Model Wizard* opens, select the room dimension in the middle panel. The default selection is 3D. Then click the *Next* button.
- 3 On the *Add Physics* panel select **Heat Transfer** → **Electromagnetic Heating** → **Joule Heating: right button** → **Add Selected**. Then click the *Next* button (Fig. 4).
- 4 Select *Stationary* on the *Select Study Type* panel. Then click the *Finish* button (Fig. 5).

## Global Definitions

The *Global Definitions* branch is the place to define the global parameters, what are active for the whole simulation. It contains sections like *Parameters*, *Variables*, and *Functions*.

- 1 **Global Definitions: right button** → **Parameters**. Click on the first cell in the *Name* Column and enter *L*(Fig. 6).
- 2 Click on the first cell of the *Expressions* column and enter value for *L*: **9 [cm]**. The angle brackets make possible to enter arbitrary units.
- 3 Fill in the parameters table according to the Table 1.

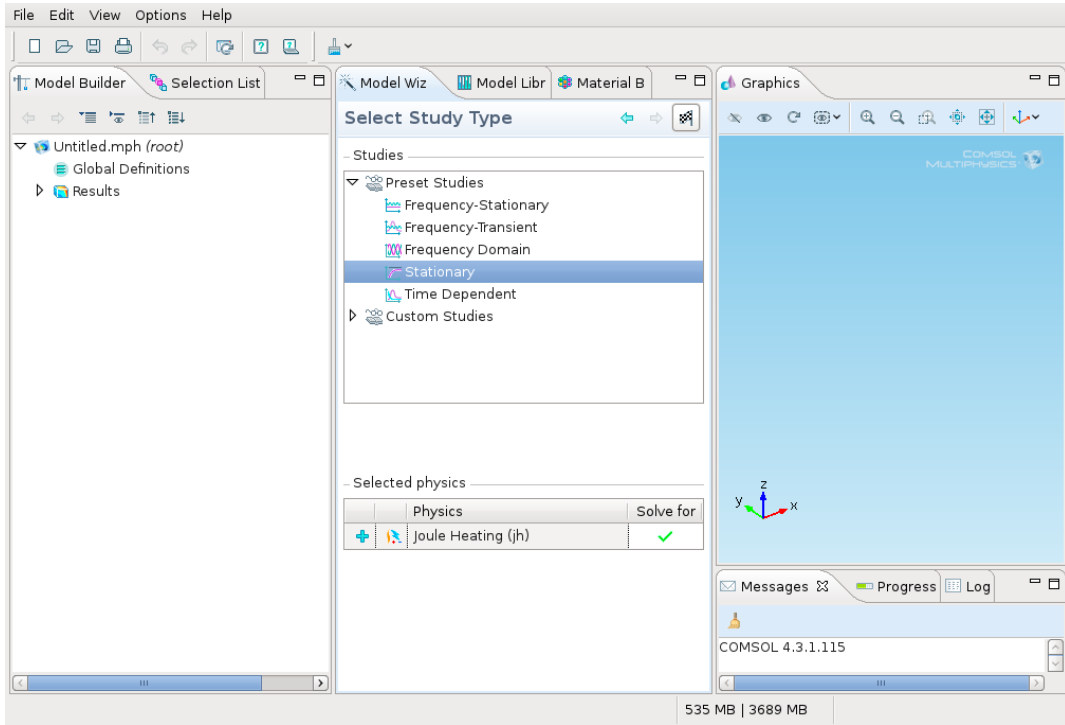


Figure 5: Selecting the type of the simulation.

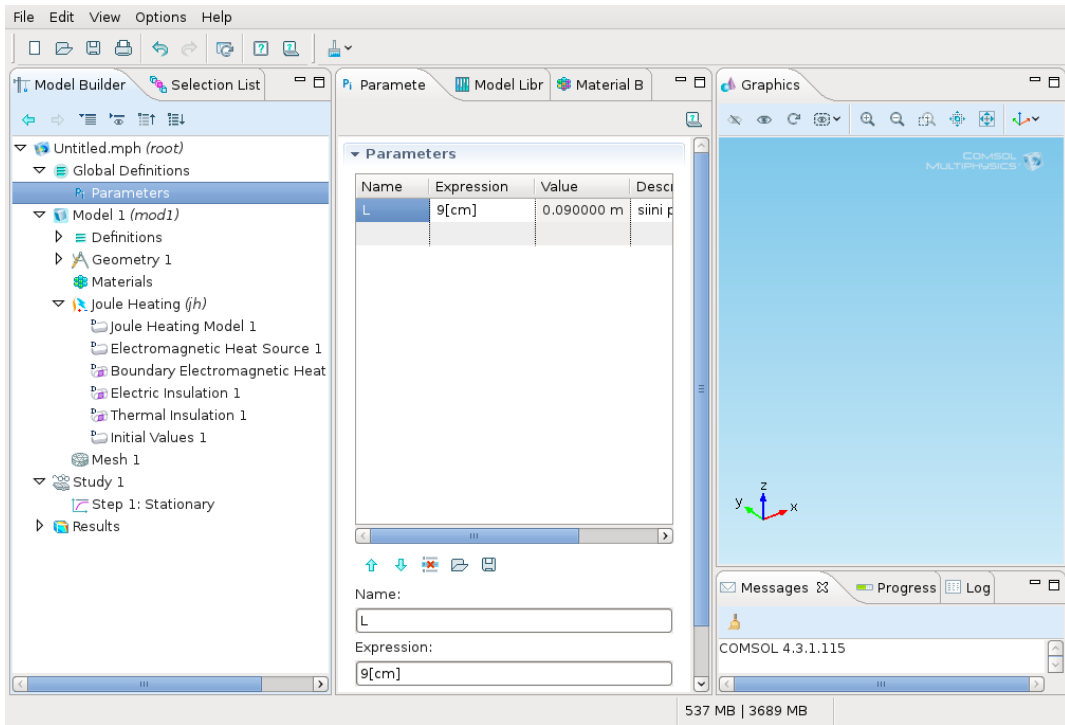


Figure 6: The table of the global parameters.

L	9[cm]	0.090000 m	Busbar length
rad_1	6[mm]	0.0060000 m	Polt radius
tbb	5[mm]	0.0050000 m	Busbar thickness
wbb	5[cm]	0.050000 m	Busbar width
mh	6[mm]	0.0060000 m	Mesh control
htc	5[W/m <sup>2</sup> /K]	5.0000 W/(m <sup>2</sup> ·K)	Heat transfer coefficient
Vtot	20[mV]	0.0200000 V	Electric potential on the bottom surface of the titan polt

Table 1: Global input parameters.

## Creating the Geometry

The present section describes step-by-step the drawing of the geometry with *COMSOL Multiphysics* tools. The parameters are described in the *Global Definitions* section. Parametrised dimensions allow the use of the conditional analysis and automated parametric constraints.

1 The first step is to draw a profile of the busbar:

- **Model 1: right button → Geometry 1 → Work Plane.** *Work Plane* parameters set as in the Fig. 7;
- **Plane → xyz;**
- Click the **Show Work Plane** button in the parameters button bar.

2 Next set parameters for the axes and the grid:

**Model Builder → View 2 → Axis** (Fig. 8):

(a) for the axes:

- ***x minimum = -0.01;***
- ***y minimum = -0.01;***
- ***x maximum = 0.11;***
- ***y maximum = 0.11;***

(b) for the grid:

- select ***Manual Spacing;***
- ***x spacing = y spacing = 5e-3.***

3 Click the **Apply** button.

4 There are at least two ways to draw the geometry:

- right click on the **Geometry** or **Plane Geometry** branch to add a geometrical object;
- click on the **Plane Gometry** branch under **Work Plane** to open the interactive drawing bar.

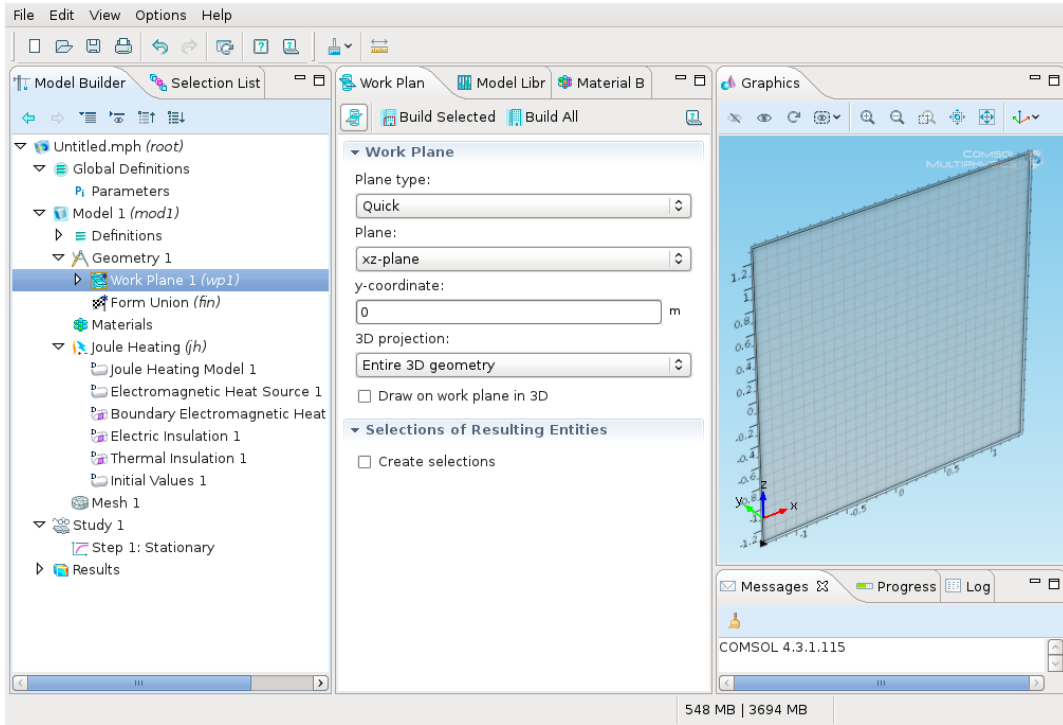


Figure 7: Work plane.

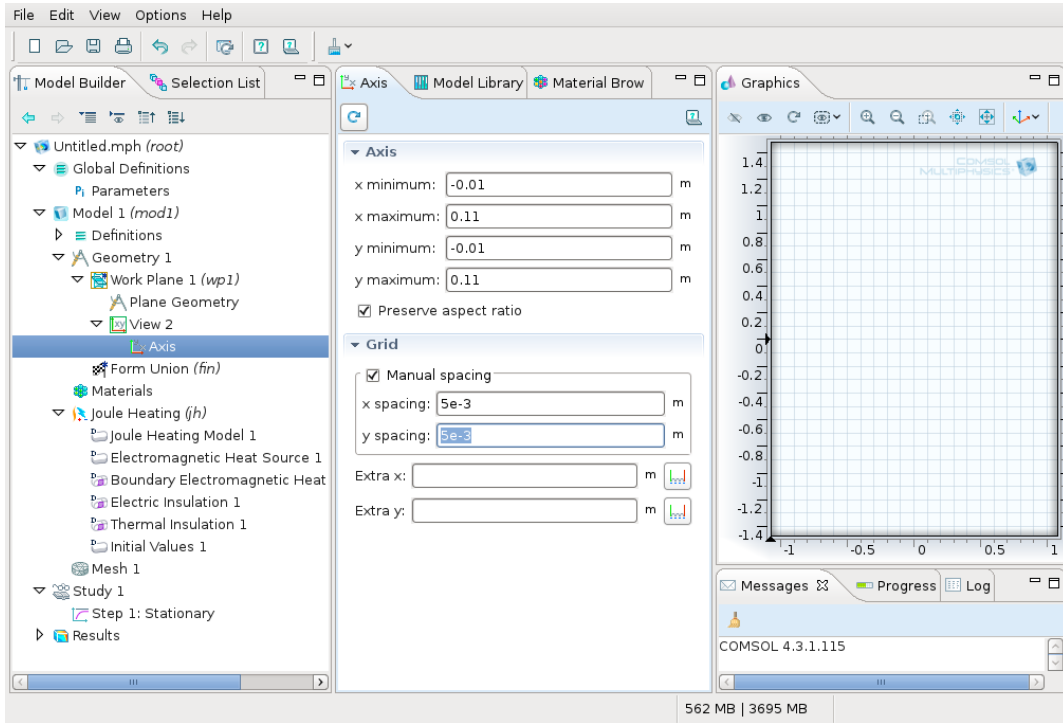


Figure 8: Axes parameters.

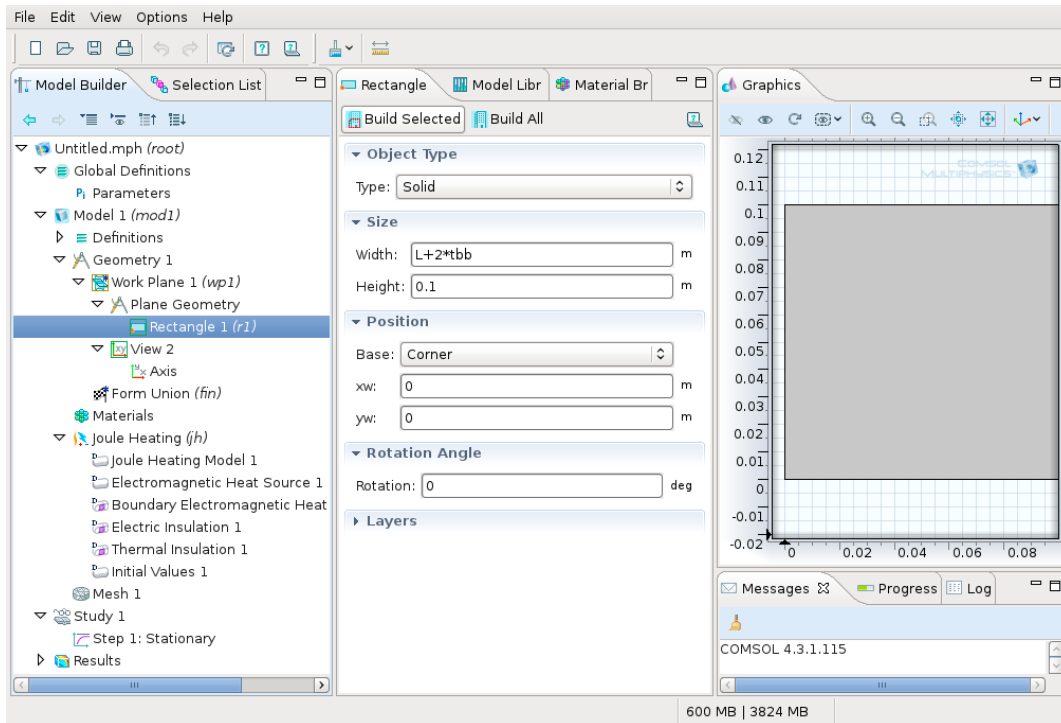


Figure 9: Drawing the rectangle.

Using the first method:

***Model Builder*** → ***Work Plane 1*** → ***Plane Geometry: right button*** → ***Rectangle***. Rectangle parameters (Fig. 9):

- ***Width =  $L_2 * tbb$*** ;
- ***Height = 0.1***.

Click the ***Build Selected*** button.

5 Draw another rectangle:

- ***Width =  $L + tbb$*** ;
- ***Height =  $0.1 - tbb$*** ;
- ***Position:  $yw = tbb$*** .

Click the ***Build Selected*** button.

6 To make the selection of geometrical entities more easier switch on their labels:

***Model Builder*** → ***Geometry 1*** → ***Work Plane*** → ***Plane Geometry*** → ***View 2: Show geometry labels = v*** (Fig. 10).

7 To subtract the second rectangle from the first:

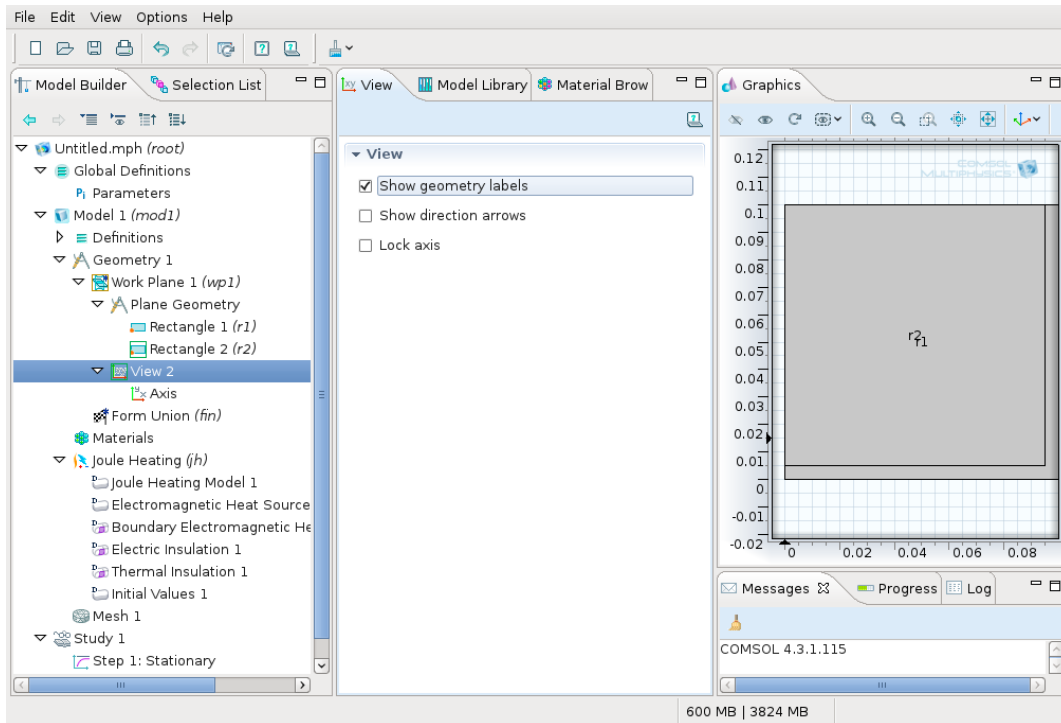


Figure 10: Showing the labels.

- select  $r1$  as a minuend: **Work Plane 1** → **Plane Geometry: right button** → **Boolean Operations** → **Difference**;
- click on the label  $r1$  in the rendering window. The selection turns red.
- then click the right button outside the selected geometry and the selected geometrical entity turns blue marking the successful selection (Fig. 11);
- in the **Difference** window click the **Activate Selection** button right from the **Objects to subtract** window;
- after that use the method described above to select smaller rectangle  $r2$  as a subtrahend (Fig. 12);
- another way to select  $r2$  is to use the *Select List* window:

**Select View** → **Select List** →  $r2$  (solid) → **right button:  $r2$  (solid)** (Fig. 13);

- The last step is to click the **Build Selected** button (Fig. 14).

## 8 Rounding the internal corner of the L-profile:

- **Work Plane 1** → **Plane Geometry: right button** → **Fillet**
- add the point 3 into the *Vertices to fillet* list. There are several ways for that:

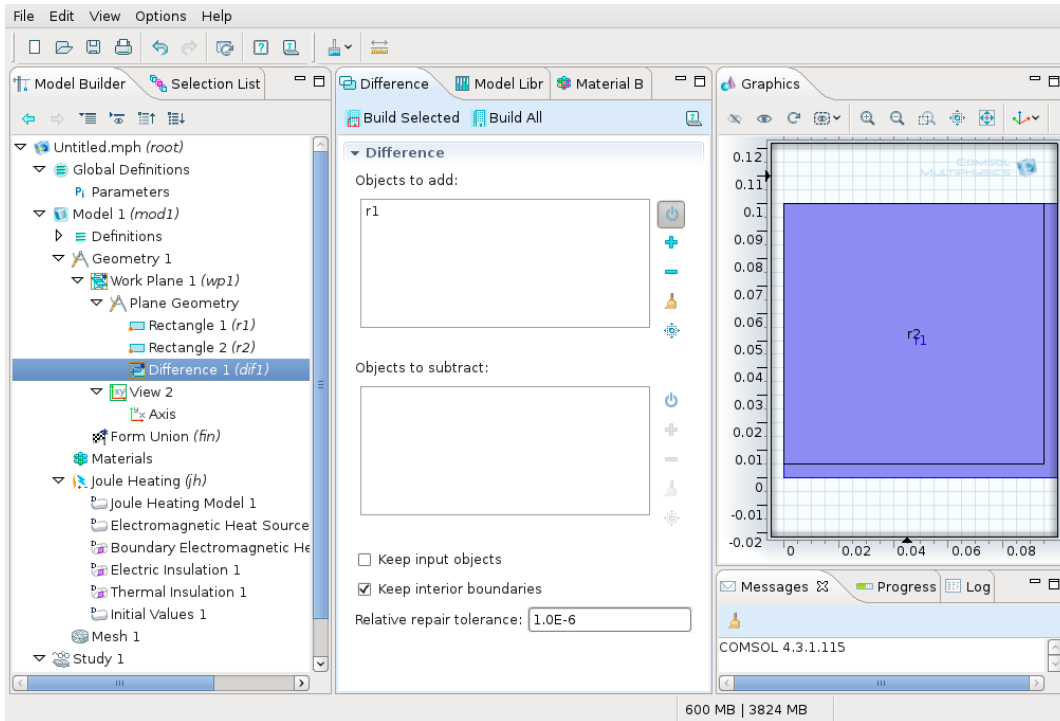


Figure 11: The minuend.

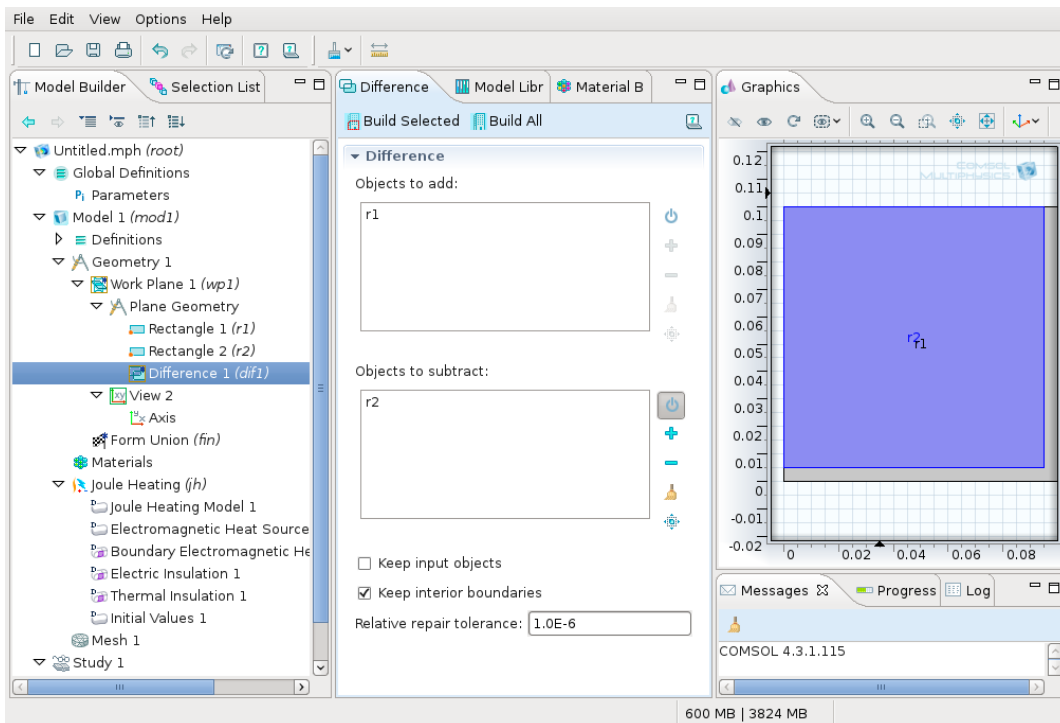


Figure 12: The subtrahend.

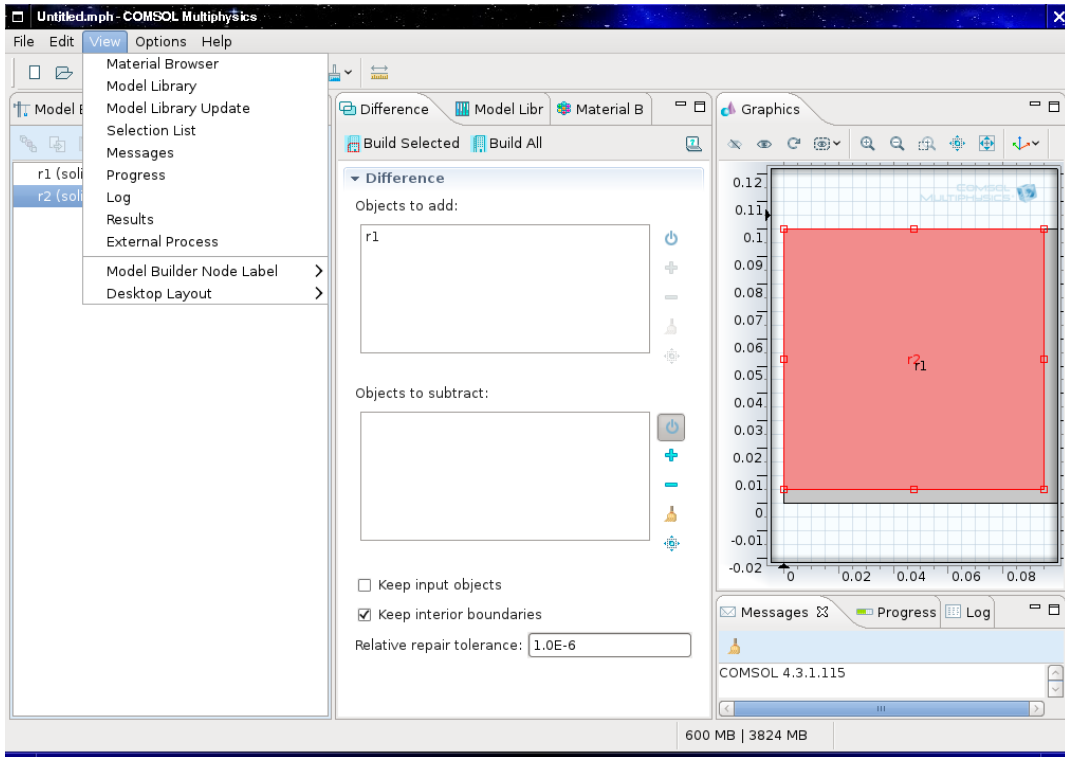


Figure 13: Subtraction of the objects.

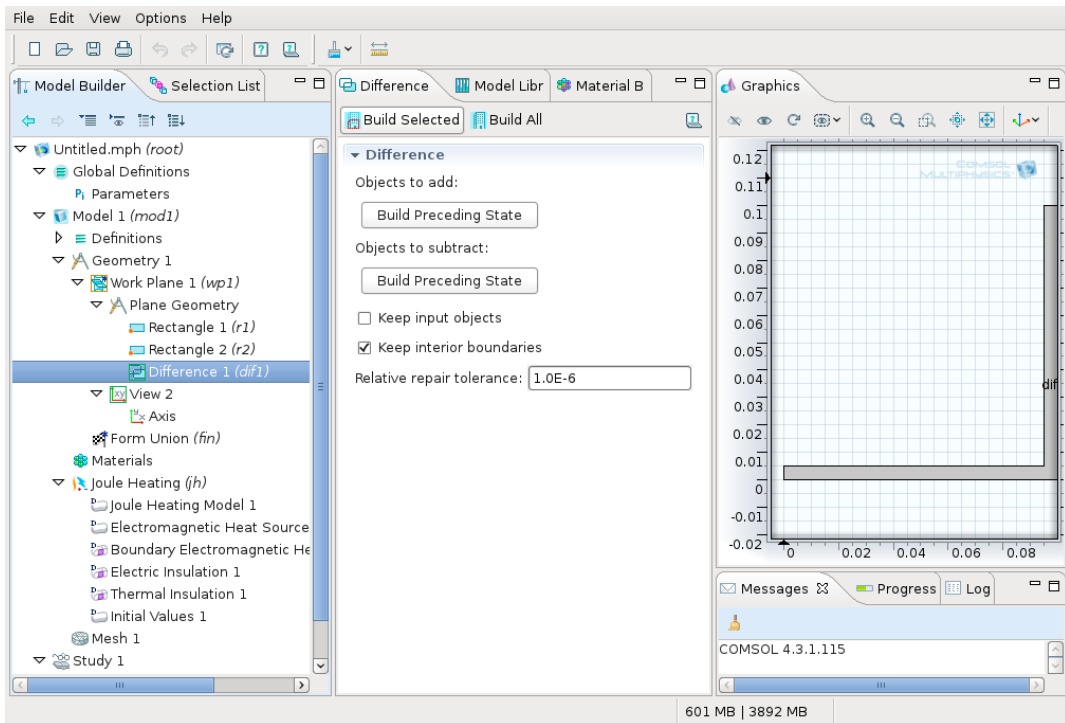


Figure 14: The difference.



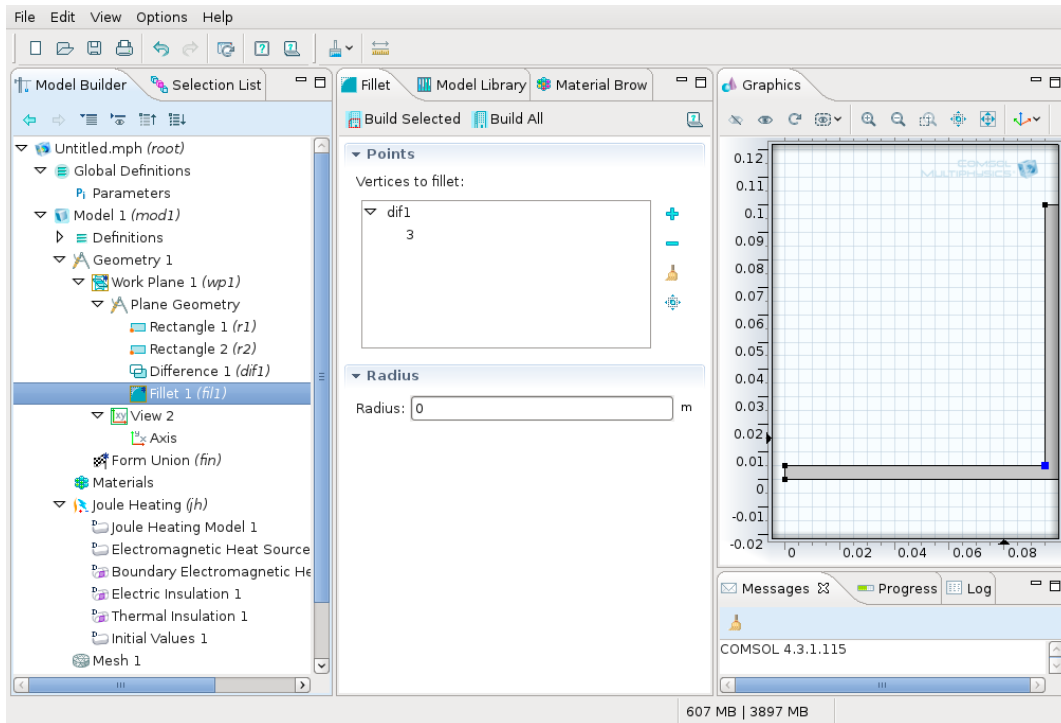


Figure 15: Selecting the point in the rendering window.

- in the rendering window, click on the appropriate point with the left mouse button turning the point red and then click the right mouse button anywhere in the rendering window turning the selected point blue (Fig. 15);
- **View** → **Selection List** → **3**, click the **Add Selection** button next to the *Vertices to fillet* window or right click in the **Selection List** (Fig. 16).

- **Radius =  $tbb$** ;
- click the **Build Selected** button (Fig. 17).

9 For the external corner:

- **Work Plane 1** → **Plane Geometry: right button** → **Fillet**;
- select the point **6** in the rendering window and add it into the *Vertices to fillet* list;
- **Radius =  $2*tbb$** ;
- click the **Build Selected** button (Fig. 18).

10 Next extrude the 2D work plane into 3D:

- **Model Builder** → **Work Plane 1: right button** → **Extrude**;
- enter the value  **$wbb$**  into the *Distances from Plane* table replacing the default value to extrude the profile to its thickness (Fig. 19).

The given table makes possible to enter several consecutive values to create a layered structure of different materials.

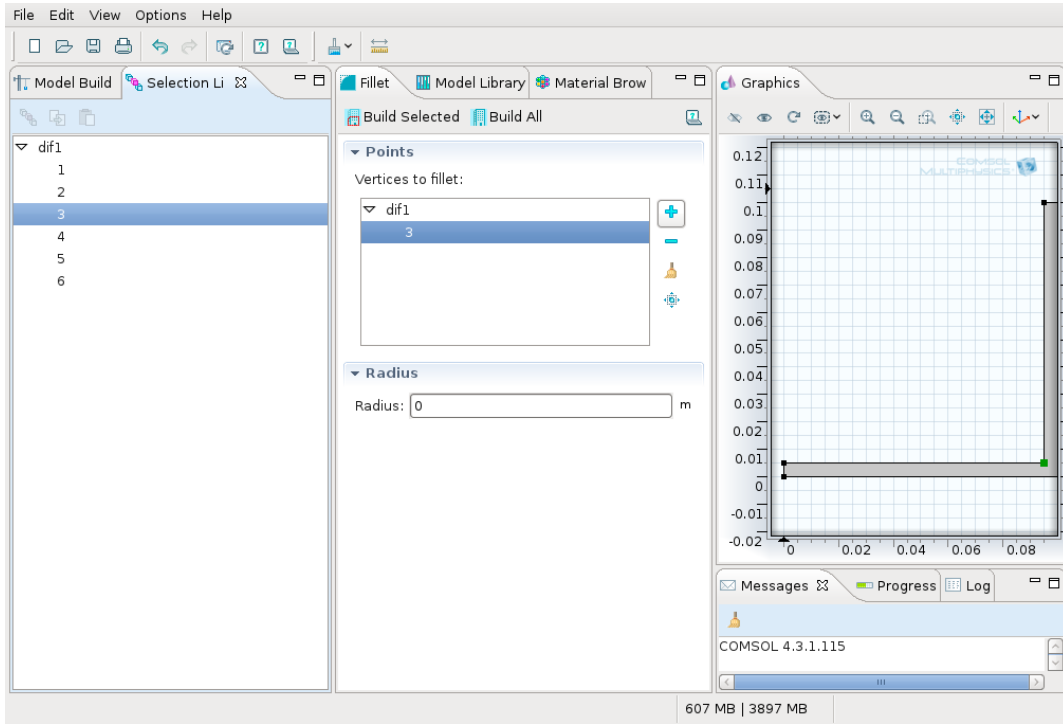


Figure 16: Selecting the point from the selection list.

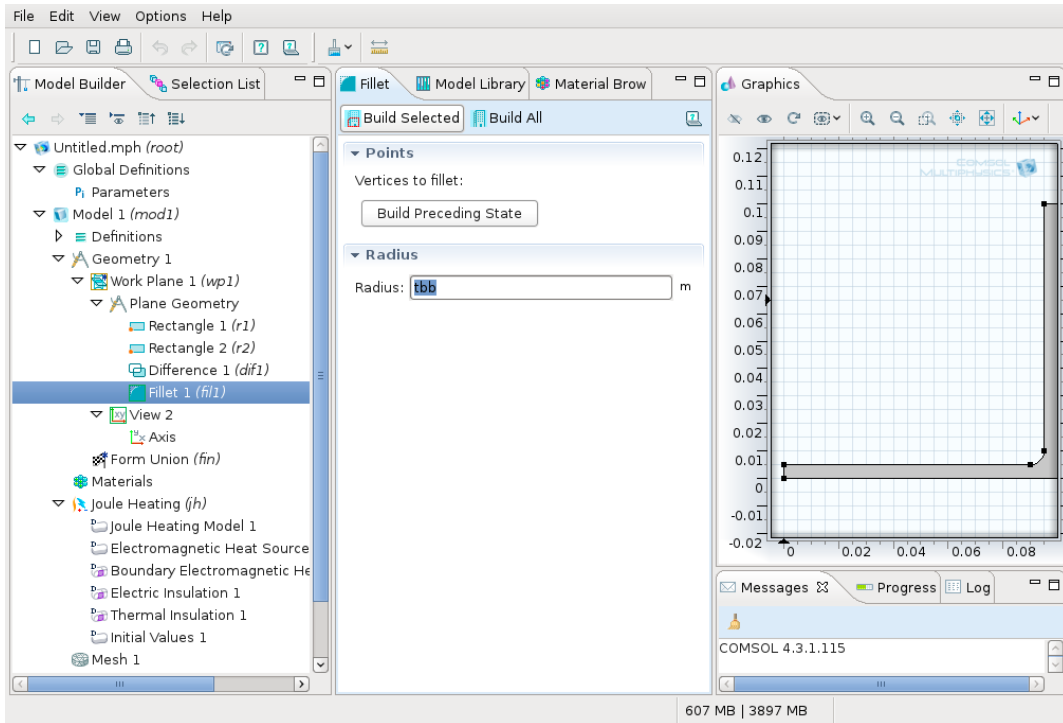


Figure 17: Rounding the internal corner.

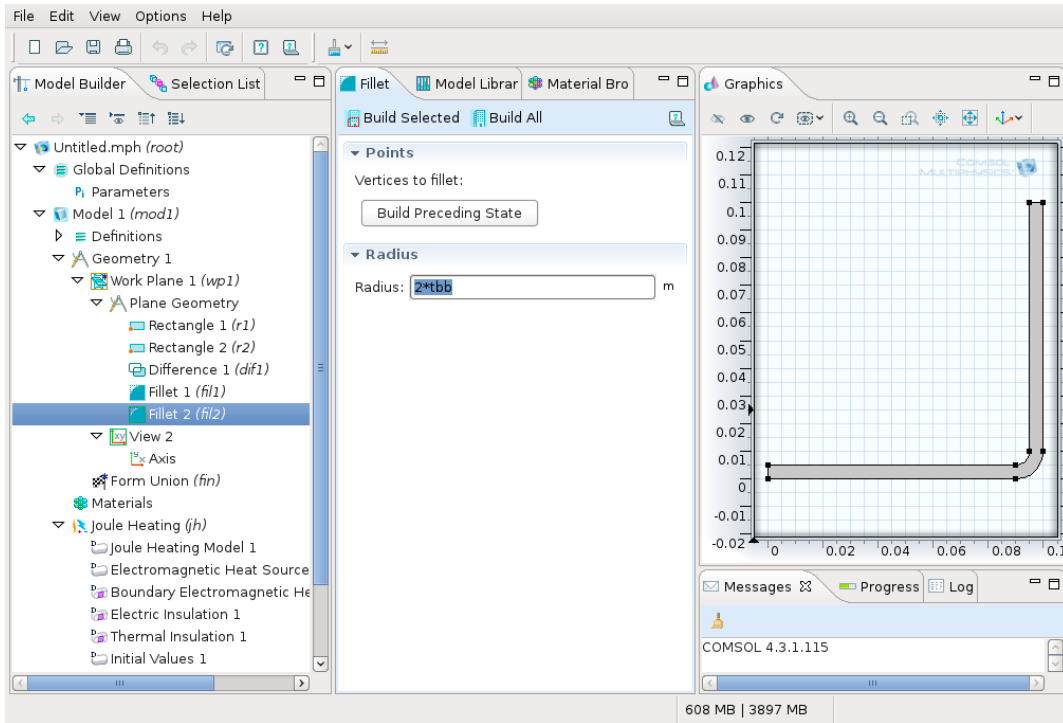


Figure 18: Rounding the external corner.

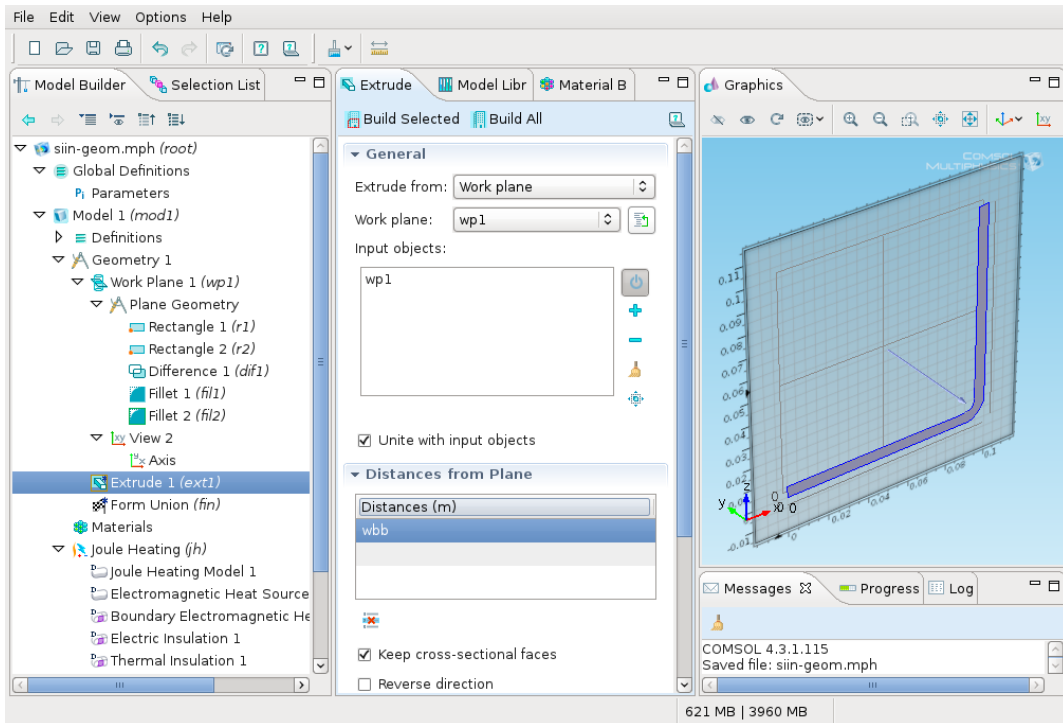


Figure 19: Extruding the work plane.

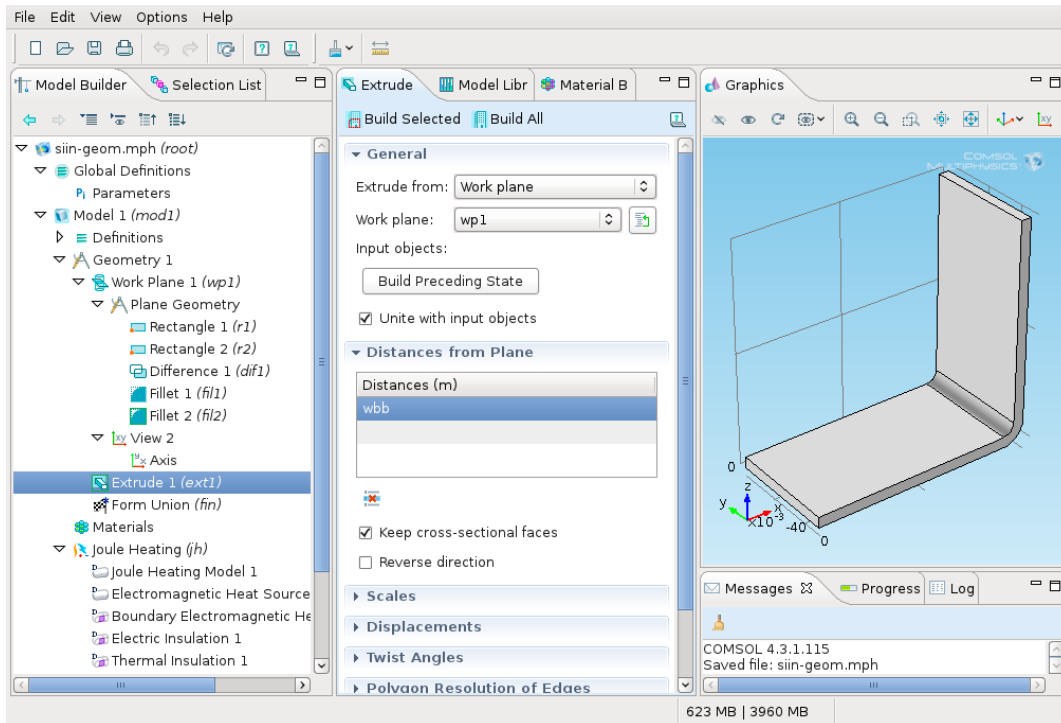


Figure 20: Bushbar geometry.

11 Click the **Build Selected** and then the **Zoom Extents** buttons. Save the model as **busbar-geometry.mph**. The resulting structure is in the Fig. 20.

12 Next construct the titan polts by extruding a circle on the new work plane:

- **Model Builder** → **Geometry 1: right button** → **Work Plane**. The work plane *Work Plane 2* is added;
- make a modification in its parameters window:  
**Work Plane** → **Plane Type = Face parallel** (Fig. 21).
- select the surface 8 in the rendering window and add it into the *Planar face* list;
- click the **Show Work Plane** button to draw the circle marking the location of the first polt. Click the **Zoom Extents** button;
- **Work Plane 2** → **Plane Geometry: right button** → **Circle**. Parameters of the circle (Fig. 22):
  - **Radius = rad\_1**.
- Click the **Build Selected** button.
- Continue the constructing of the polt by extruding it (Fig. 23):
  - **Model Builder** → **Work Plane 2: right button** → **Extrude**;
  - enter  $-2*tb$  into the first row of the *Distances from Plane*, rising the polt out of the bushbar surface by the bushbar's thickness.

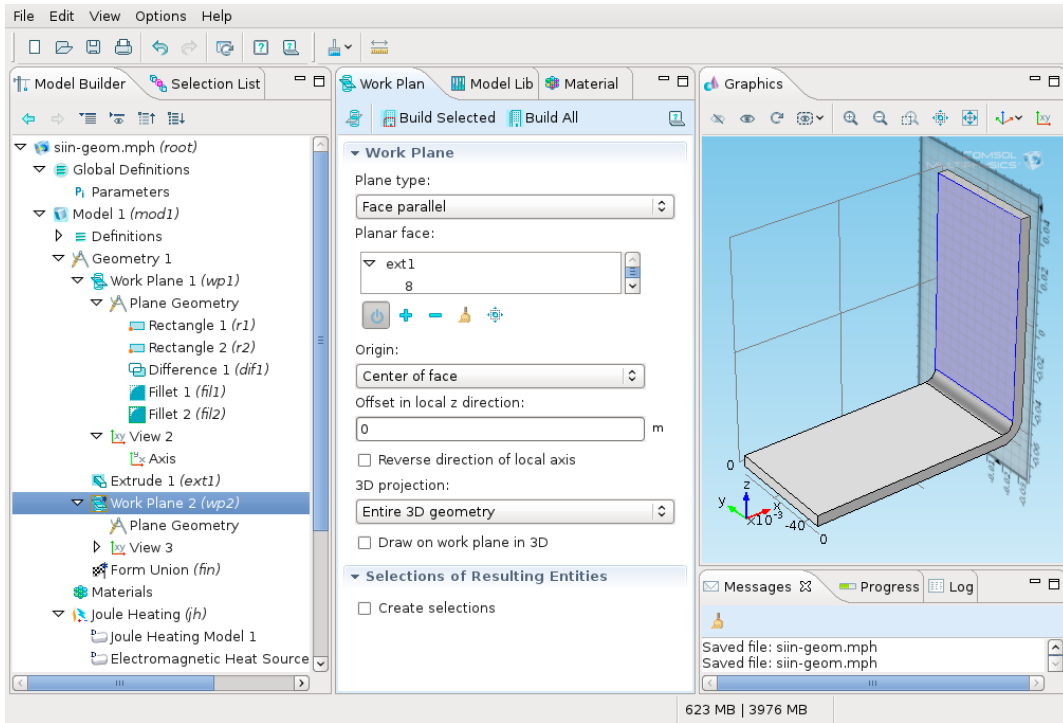


Figure 21: Work plane of the titan polts.

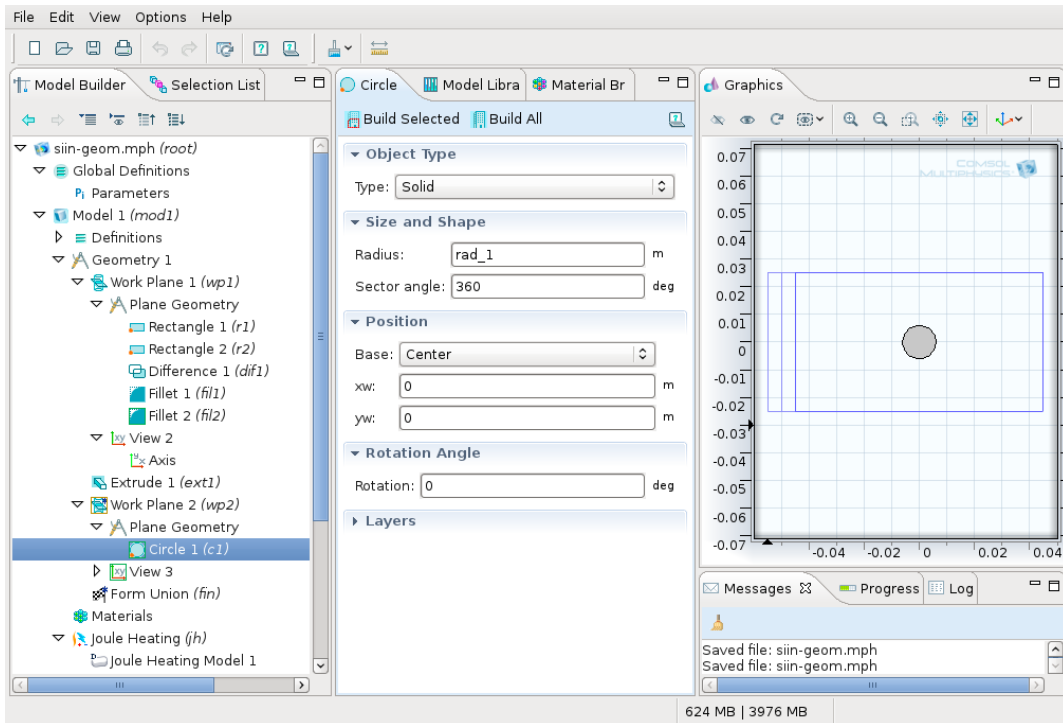


Figure 22: Drawing the polt on the work plane.

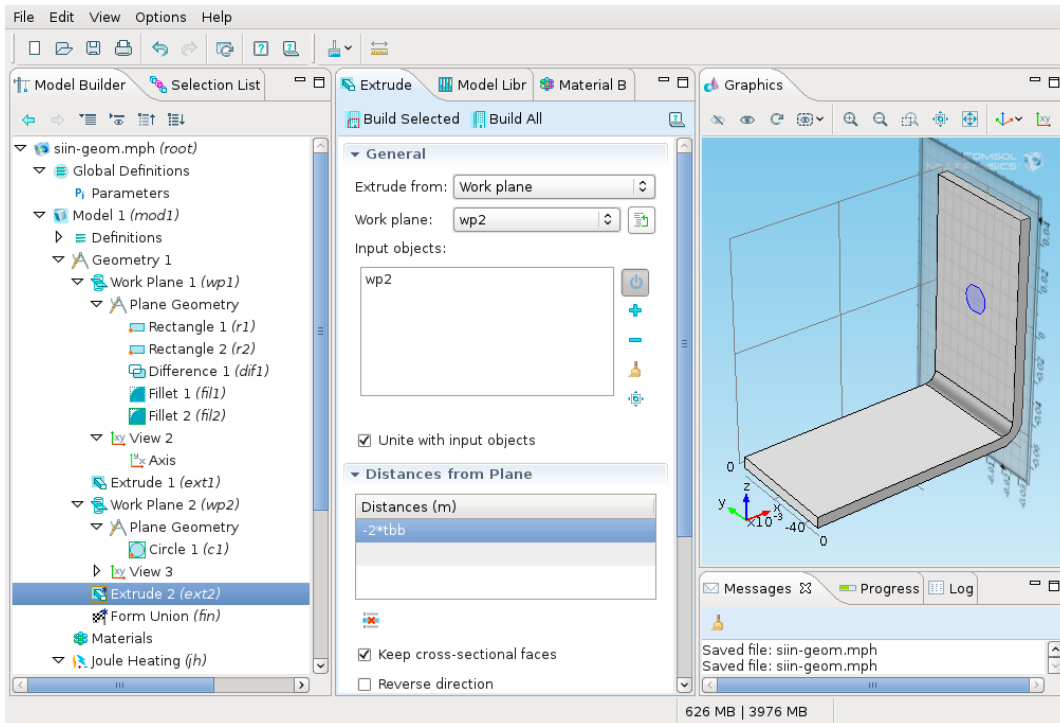


Figure 23: Extruding the polt.

- Constructing the another polt:
  - **Geometry 1: right button** → **Work Plane** → **Plane type = Face parallel** (Fig. 24);
  - add the surface **4** into the *Planar face* list;
  - click the **Show Work Plane** button and then the **Zoom Extents** button to get a better overview of the geometry;
  - **Work Plane 3** → **Plane Geometry: right button** → **Circle** (Fig. 25):
    - \*  $Radius = rad\_1$ ;
    - \*  $xw = -L/2 + 1.5e-2$ ;
    - \*  $yw = -wbb/4$ .
  - Click the **Build Selected** button.
- The third polt is constructed by copying the second polt:
  - **Work Plane 3** → **Plane Geometry: right button** → **Transforms** → **Copy** (Fig. 26);
  - click on the **c1** in the rendering window and then right-click anywhere else in the rendering window to add the circle **Copy 1** into the *Input objects* list in the parameters window;
  - in the same parameters window set  $wbb/2$  as a value for  $yw$  in the **Displacement** section;
  - click the **Build Selected** button.

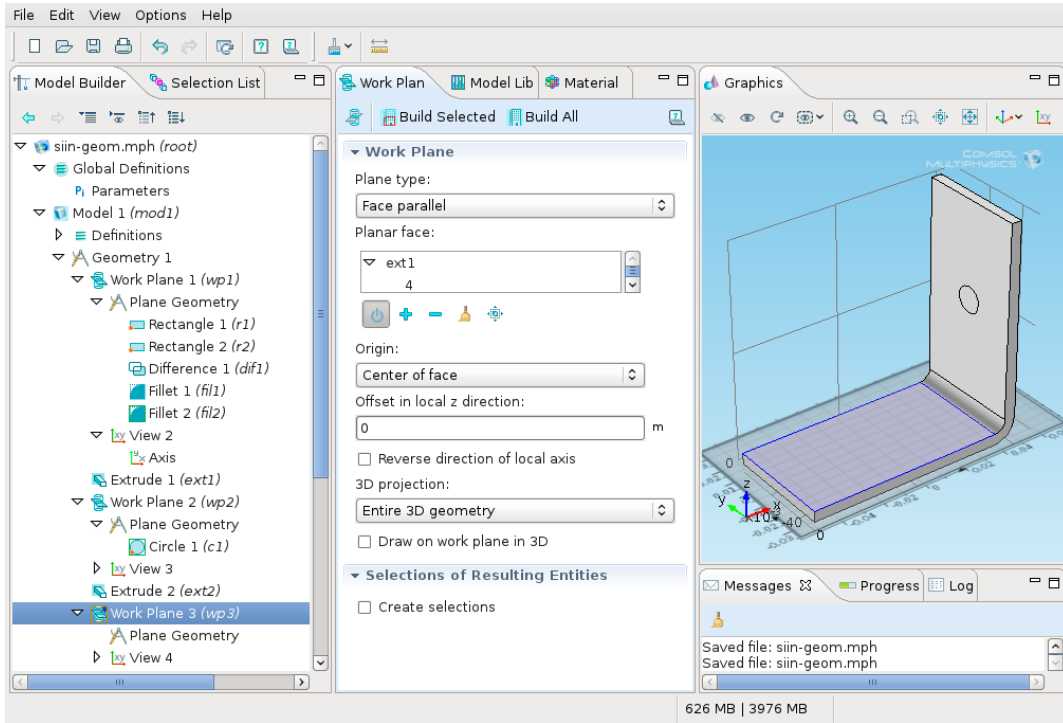


Figure 24: Adding the work plane of the second polt.

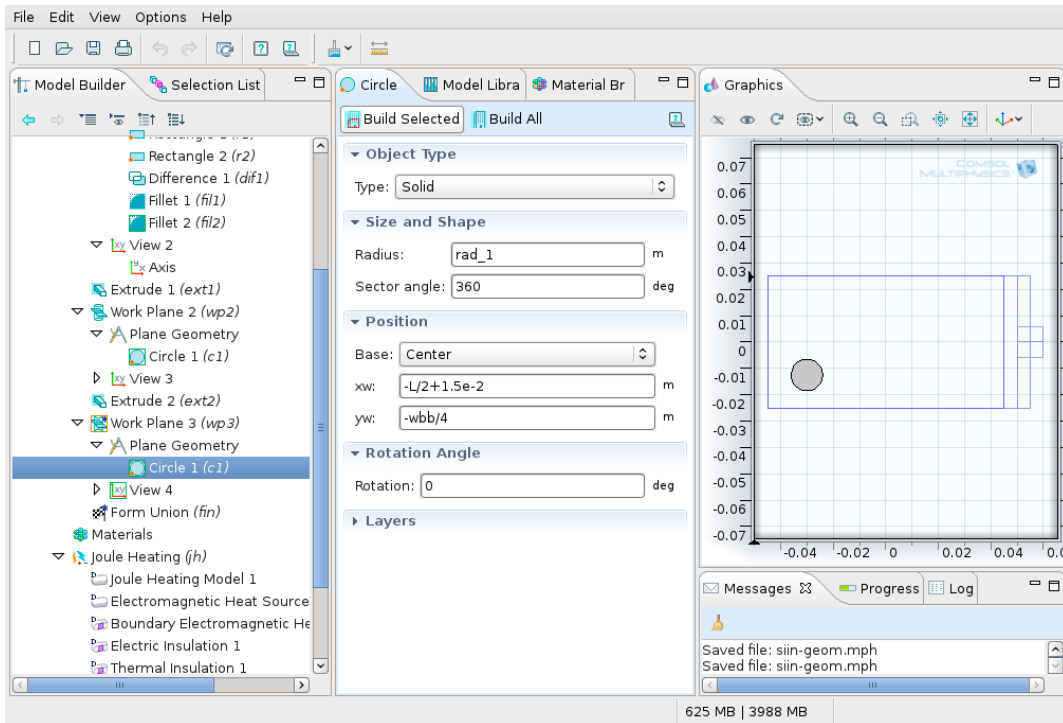


Figure 25: Drawing the second polt.

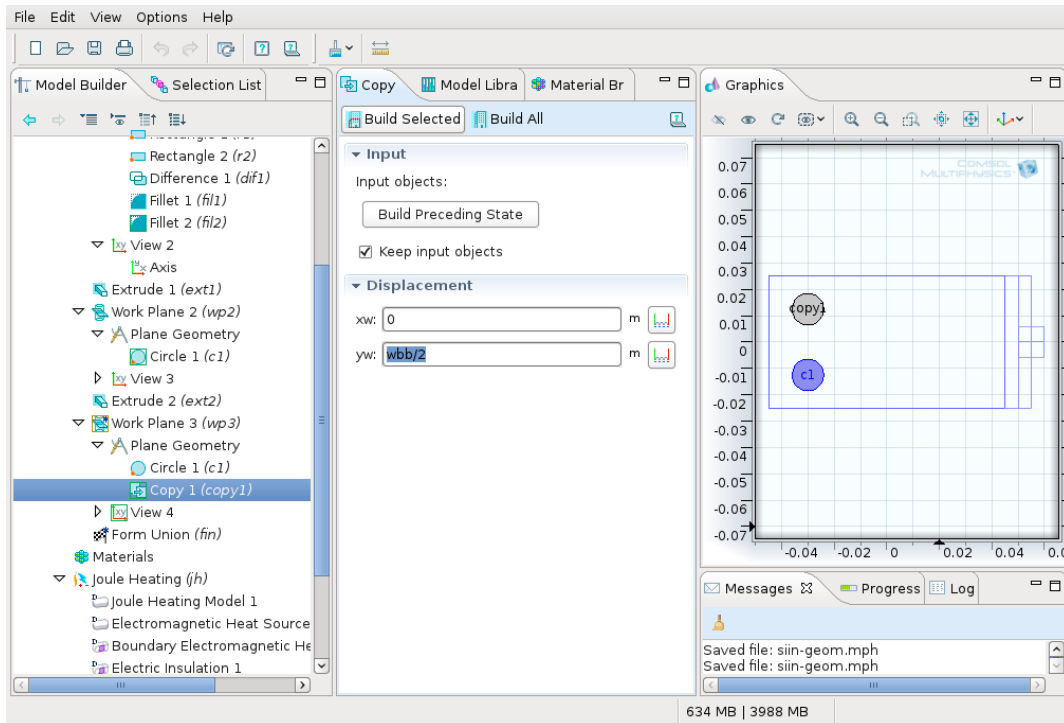


Figure 26: Drawing the third polt.

- Extrude the polts (Fig. 27):
  - **Model Builder** → **Work Plane 3: right button** → **Extrude**;
  - replace the default value in the first row of the *Distances from Plane* by  $-2*tbb$  in the **Extrude** parameter window;

**13** Click the **Build All** button.

**14** Save the file as **bushbar-geometry.mph**.



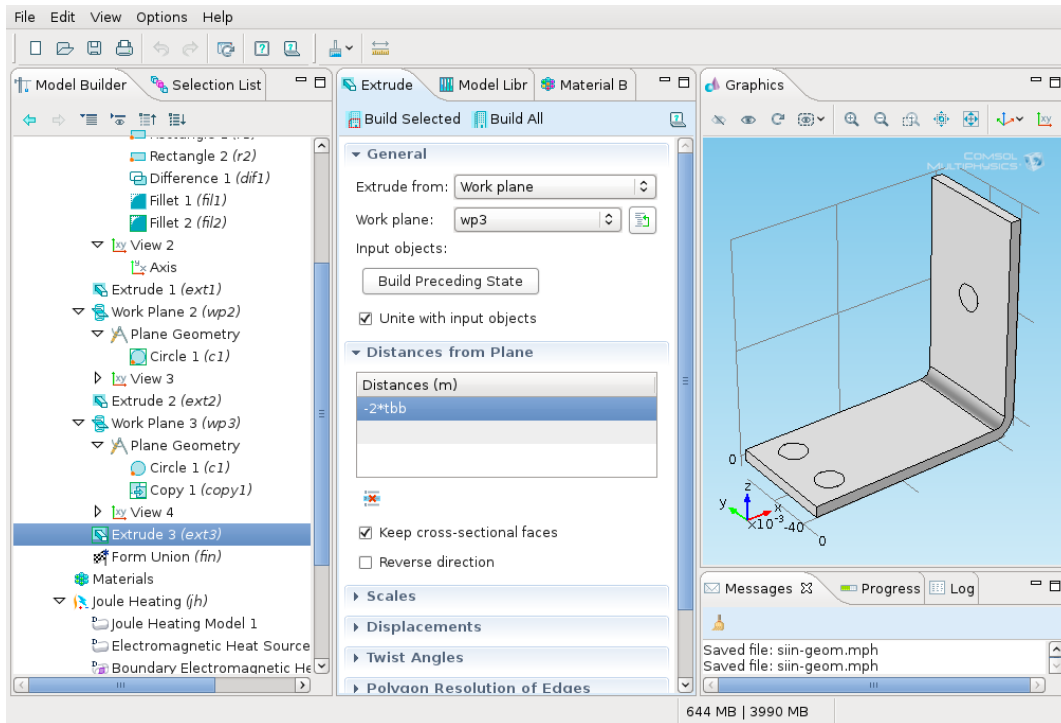


Figure 27: Extruding the polts.

---

### Task 1:

- 1 Plot the default 3D view of the busbar geometry.
  - 2 Measure the value of the  $wbb$  in the rendering window and plot the value along with the selected measuring points.
  - 3 Measure the value of the  $L$  in the rendering window and plot the value along with the selected measuring points.
  - 4 Measure the value of the  $tbb$  in the rendering window and plot the value along with the selected measuring points.
  - 5 Measure the value of the  $2xrad\_1$  of the titan polt 2b and plot the value along with the selected measuring points.
- 

## Importing the Geometry from the File

The last chapter gave quite a detail instructions how to create the geometry using the *COMSOL Multiphysics* tools. But it is possible to import pre-generated geometry into *COMSOL Multi-*

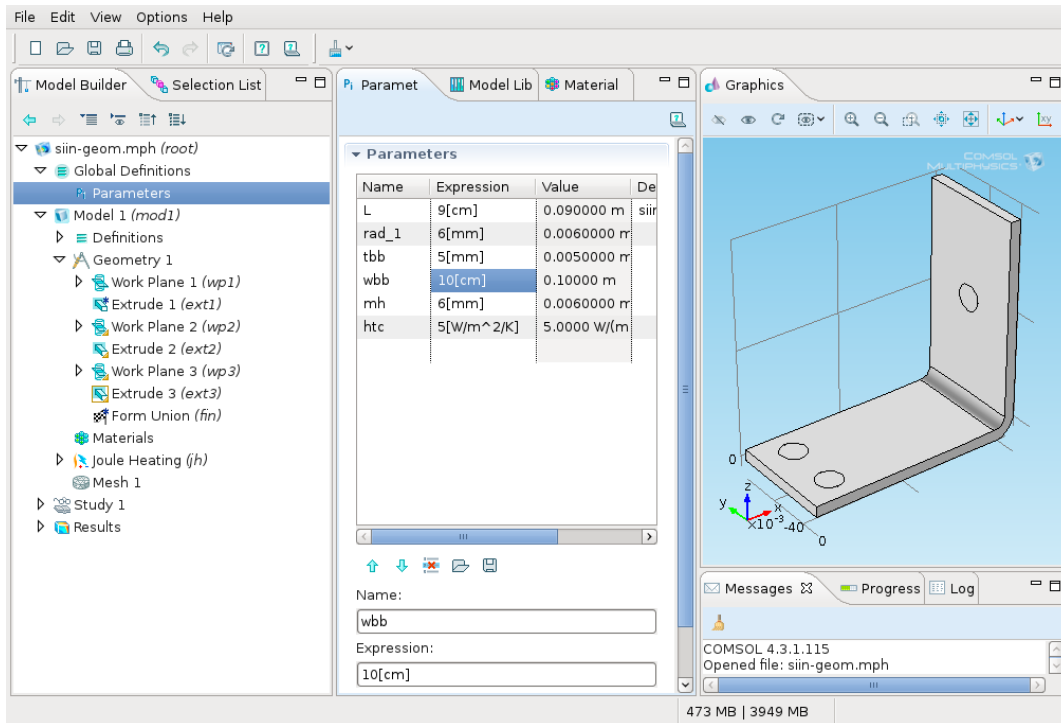


Figure 28: Changing the bushbar width.

*physics* from the file. It is suggested to start a new project by creating the geometry and saving it into the separate file. This gives a chance to restart the simulation without recreating the geometry if anything goes wrong. Besides, creating the geometry can be sometimes time-consuming. One have to keep in mind:

**the numbering of the nodes, edges, surfaces etc. may depend on the order of the creation of the geometrical objects.**

- 1 Open the file *bushbar-geometry.mph* containing the geometry: **File** → **Open** →...
- 2 **Global Definitions** → **Parameters**: set **10 cm** as a value for the **wbb** (Fig. 28).
- 3 Execute **Model Builder** → **Form Union** and then rerender the geometry by clicking the **Build All** button. The result is in the Fig. 29.
- 4 Experiment with the geometry in the rendering window:
  - rotate the busbar with the mouse holding left button down;
  - move the busbar with the mouse holding the right button down;
  - zoom the busbar with the mouse holding the middle button down;
  - restore the default view by clicking the **Go to Default 3D View** button in the rendering window;
  - restore the busbar default width: **wbb = 5 cm**.

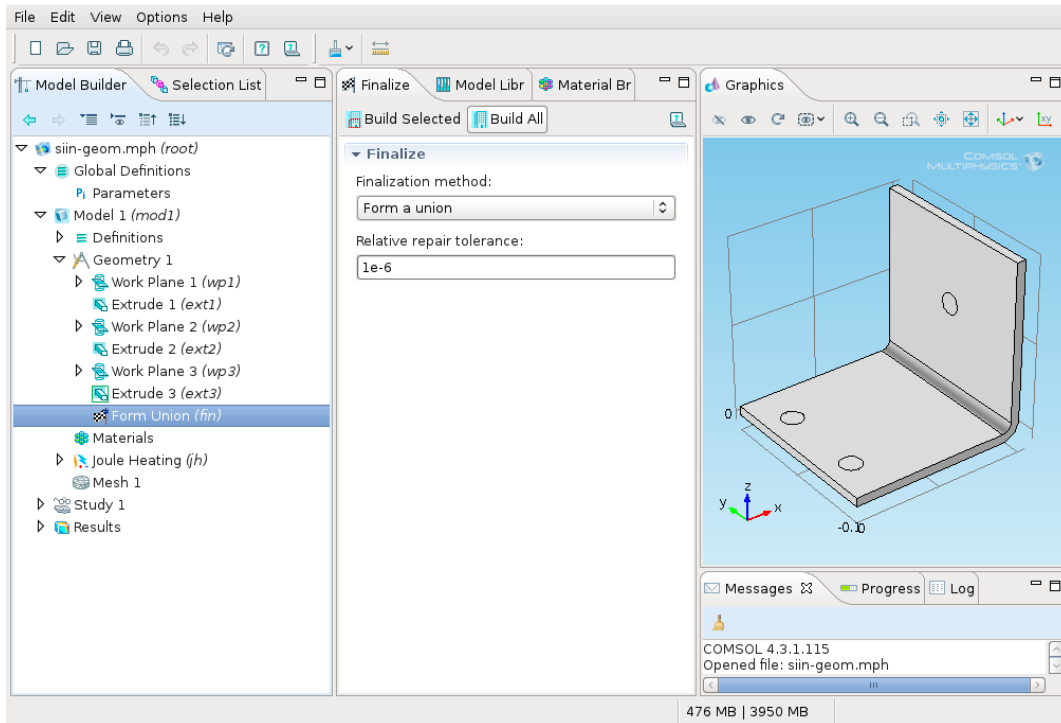


Figure 29: The busbar with the new width.

---

**Task 2:** Plot the bushbar geometry in the case of the  $wbb = 10 \text{ cm}$ .

---

## Material

**Materials** section contains the information for all physical phenomena and domains in the model. At this case, the busbar consists of copper and the polts of titan. Parameters for copper and titan are available in the built-in **Material Library** database.

- 1 **Model Builder** → **Materials: right button** → **Open Material Browser** or **View** → **Material Browser**.
- 2 **Material Browser** → **Built-In** → **Copper: right button** → **Add Material to Model** (Fig. 30).
- 3 Add built-in material **Titanium beta-21S** to the model.

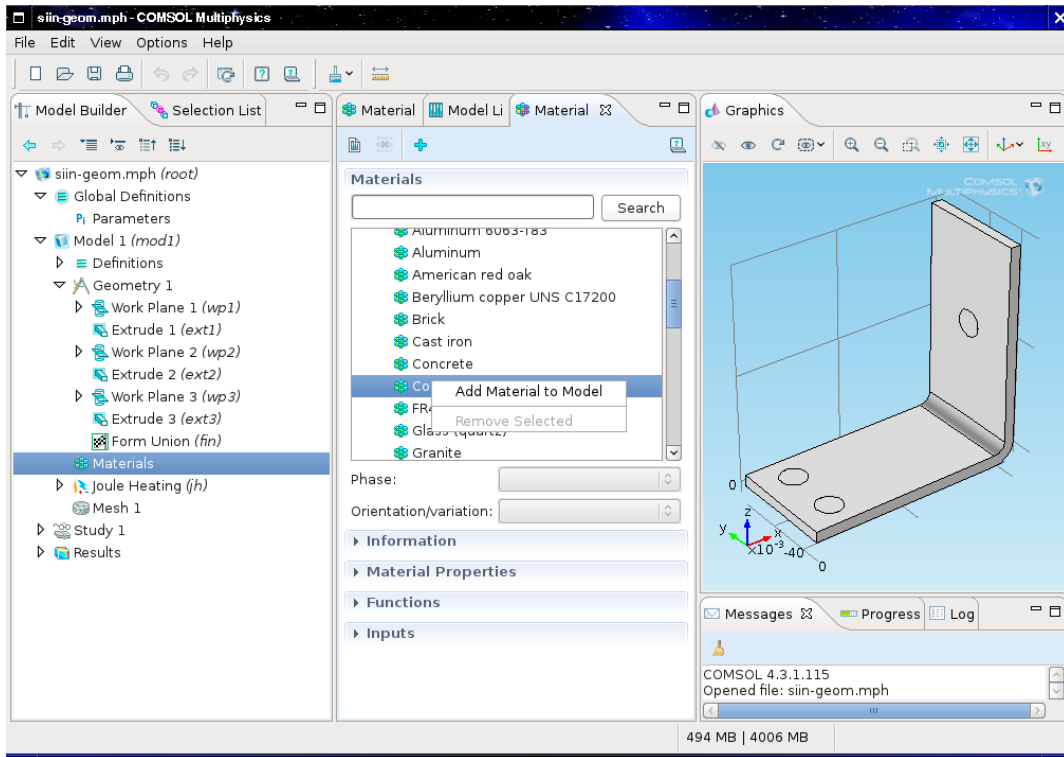


Figure 30: Adding a material.

---

**Remark:** *Material Contents* has useful information about the material properties. The properties what are needed by the physical phenomena and are available for them are marked by the green checkmark. Properties what the phenonenon needs, but missing in the material table are marked by the yellow exclamation mark. Properties, what are available but not needed have no marks. This properties may be handy later in the simulation process if the model is complemented.

---

4 Since copper was added first to the model, then all domains of the model adopt automatically its properties. With the next step, the polts are given the properties of titan what replace the properties of copper:

- **Model Builder** → **Titanum beta-21S**;
- **Selection** → **All domains**;
- now remove the domain **1**. To remove a domain or any geometrical entity like an edge, border or point:
  - select domain **1**: **Material settings** → **Selection** → **1**;
  - then click the **Remove from Selection** button;
  - another way is to click on the domain **1** in the rendering window and then remove it with the right mouse click from the **Selection** list.

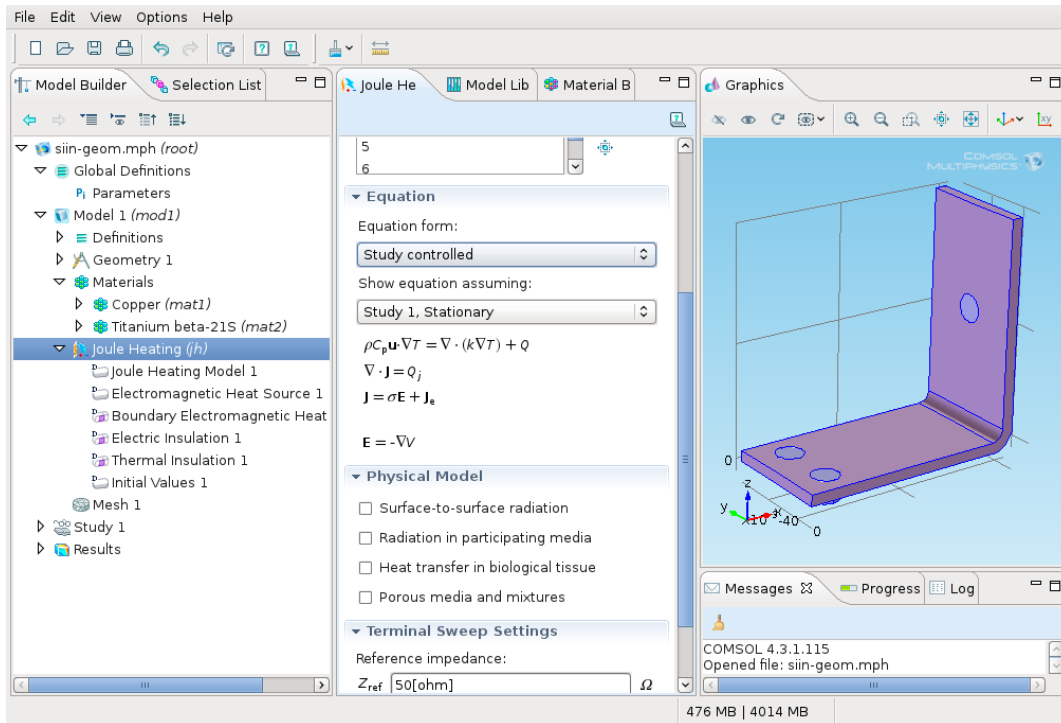


Figure 31: *Joule Heating*.

- 5 Ensure that all other domains have the properties of the titan and domain *I* the properties of copper.

## Physics

Since the introducing of the materials properties sets the domain parameters, the boundary conditions for the thermal conductivity and electrical conductivity must be set.

- While expanding the *Joule Heating* section in the *Model Builder*, the *D* in the upper left corner of the icon marks the default section.
- The equations are located in the *Equation* section and their form depends on the selection in the *Equation form* list.
- The default form of the equations depends on the simulation type specified in the *Model Wizard*. In the case of the *Joule Heating* simulations the equations are shown in the form where they are solved in respect of the temperature and electrical potential (Fig. 31).
- If one wishes to see the sections in their expanded view, one has to click the *Expand Sections* in the *Model Builder* toolbar and select for example *Equations* to display always the equations in the physics parameters window.

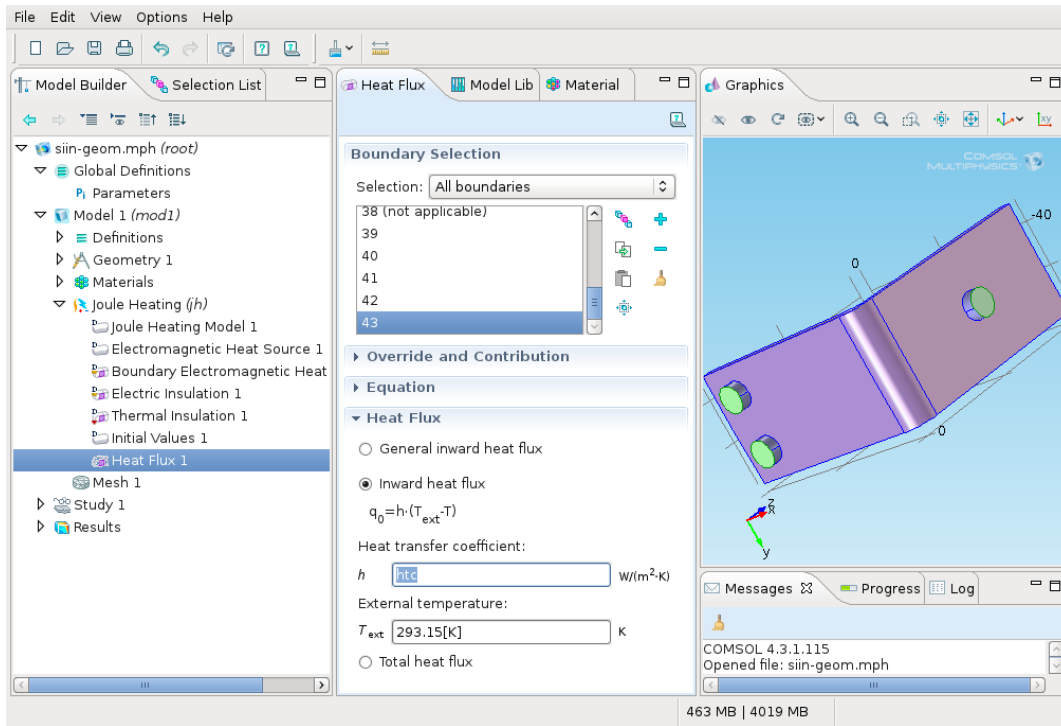


Figure 32: Boundary conditions: *Heat Flux*.

- The domain-level section *Joule Heating Model 1* contains parameters for the thermal and electrical conductivities. The contribution of the given section to the overall equation system is marked by the dotted underline in the *Equation* section.
- The thermal effect produced by the *Joule Heating* is determined in the *Electromagnetic Heat Source 1* section. The *Thermal Insulation 1* contains the default boundary conditions for the thermal conductivity problem and the *Electric Insulation 1* corresponds to the current conservation.
- *Initial Values 1* holds initial guesses for the non-linear solver of the stationary problem and the initial conditions for the time-dependent problem.

1 *Joule Heating: right button* → *second section* → *Heat Transfer* → *Heat Flux*.

2 Select *All boundaries* from the list in the *Heat Flux 1* parameters window. Let assume that the circular boundaries of the polts are not heated nor cooled and leave them at the default insulating boundary condition.

3 Rotate the busbar to see its backside. Click on the circular bottom of the titan polt, turning it green. Click the right mouse button anywhere in the rendering window to remove this surface from the *Selection* list. Repeat the same with the other two polts. The removed surfaces are numbered as **8**, **15** and **43** (Fig. 32).

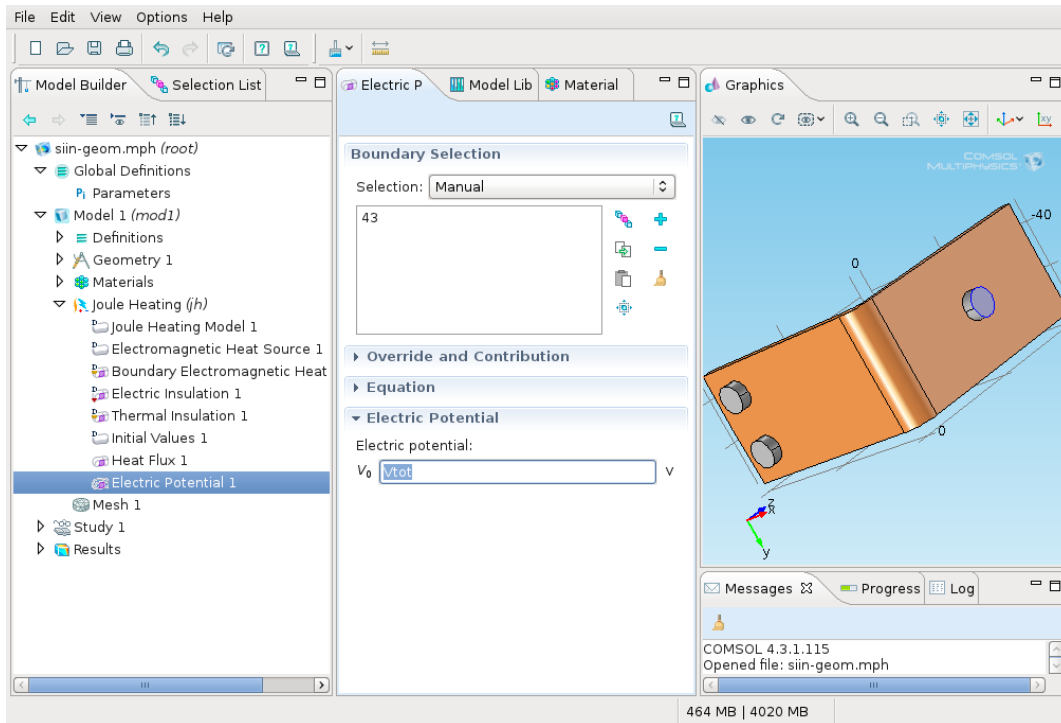


Figure 33: Boundary conditions: *Electric Potential 1*.

- 4 Click the *Inward heat flux* button in the *Heat Flux* section of the *Heat Flux* parameters window and enter *htc* as the value of the *Heat transfer coefficient h*.

This parameter is either inserted during the creation of the geometry or imported along with the geometry.

- 5 Next set the electrical current boundary conditions:

- *Model Builder* → *Joule Heating: right button* → *second section* → *Electric Currents* → *Electric Potential*;
- click on the circular bottom of the upper polt and add this boundary *43* into the *Selection* list (Fig. 33);
- enter *Vtot* as value for the *Electric potential* in the *Electric Potential* parameters window.

- 6 The last task ist to set ground boundary condition to the bottoms of the rest of the polts:

- *Model Builder* → *Joule Heating: right button* → *second section for boundary conditions* → *Electric Currents* → *Ground* (Fig. 34);
- add the boundaries *8* and *15* (the bottoms of the two remaining polts) into the *Selection* list.

- 7 Click the *Go to Default 3D View* button.

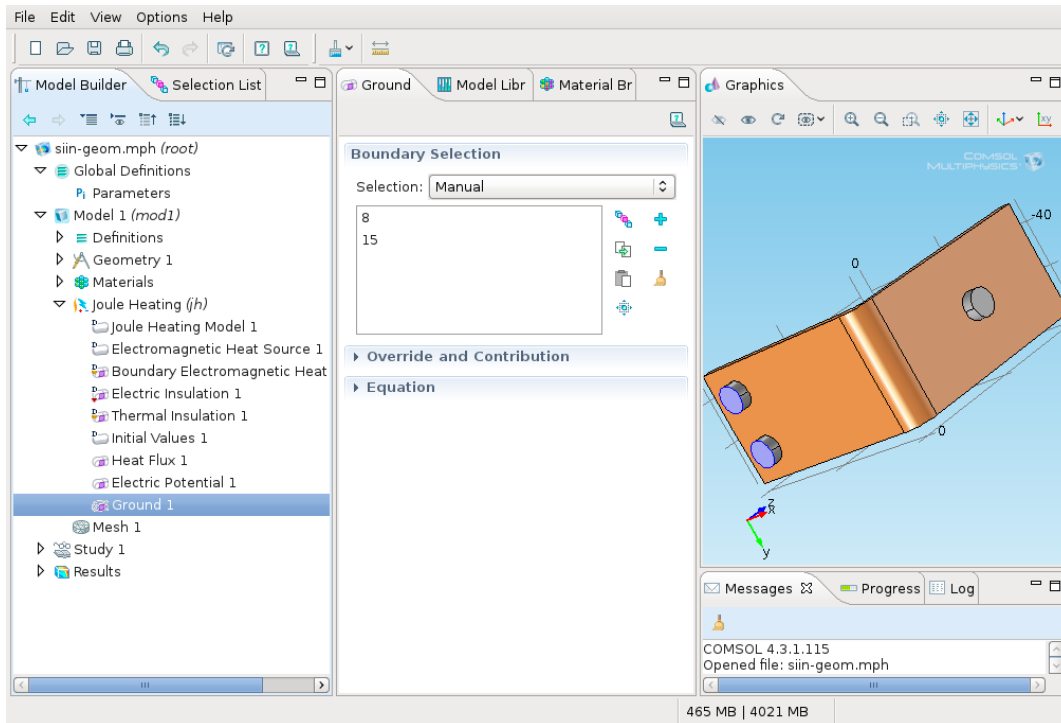


Figure 34: Boundary conditions: *Ground*.

## Mesh

The simplest mesh is an unstructured tetrahedral mesh what fits perfectly for the busbar.

---

**Remark:** The default mesh is a physics controlled mesh. Mostly, it is possible to skip the *Study* branch and just solve the model. In this example the parameters are investigated for the mesh control.

---

- 1 *Model Builder* → *Mesh 1* → *Sequence type* = *User-controlled mesh* (Fig. 35).
- 2 *Mesh 1* → *Size*. Asterisk (\*) means that the editing of the element is in progress.
- 3 Click the *Custom* button under the *Element Size* in the *Size* parameters window (Fig. 36).
  - *Maximum element size* = *mh*. Variability of the element size is restricted while using the parameter *mh*.
  - *Minimum element size* =  $mh - mh/3$ . The minimum size of the element is slightly smaller than the maximum size.
  - *Resolution of curvatures* = *0.2*. This parameter determines the number of the elements on the rounded boundaries: smaller value results in the higher resolution of the mesh along the rounded boundary. The *Maximum element growth rate* determines the element



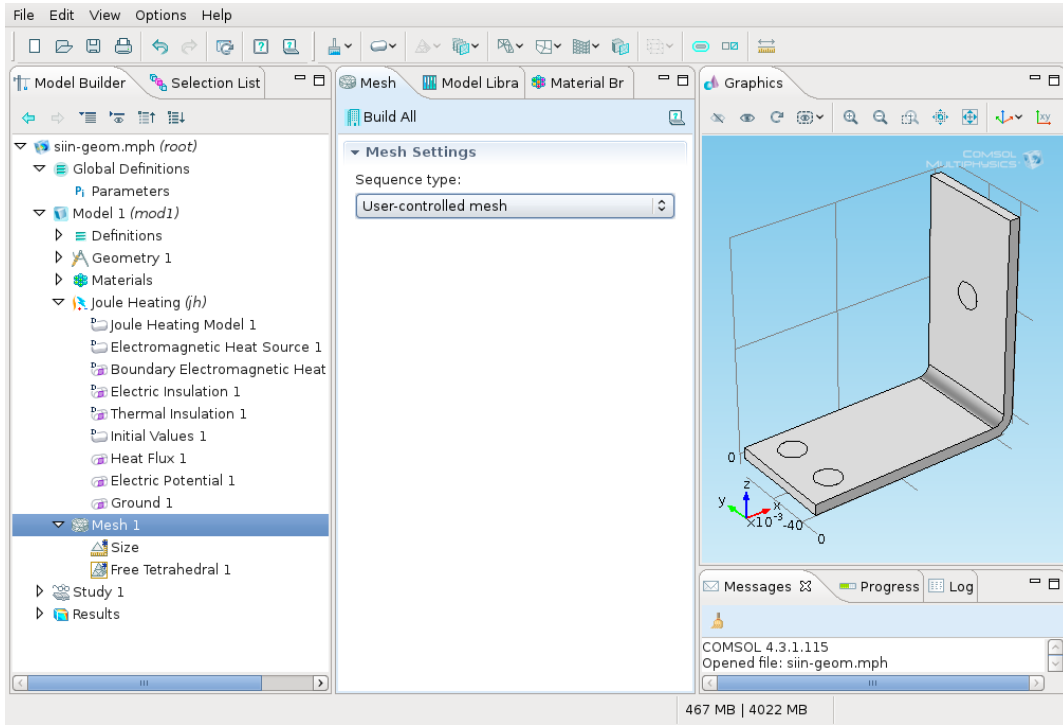


Figure 35: Controlling the mesh.

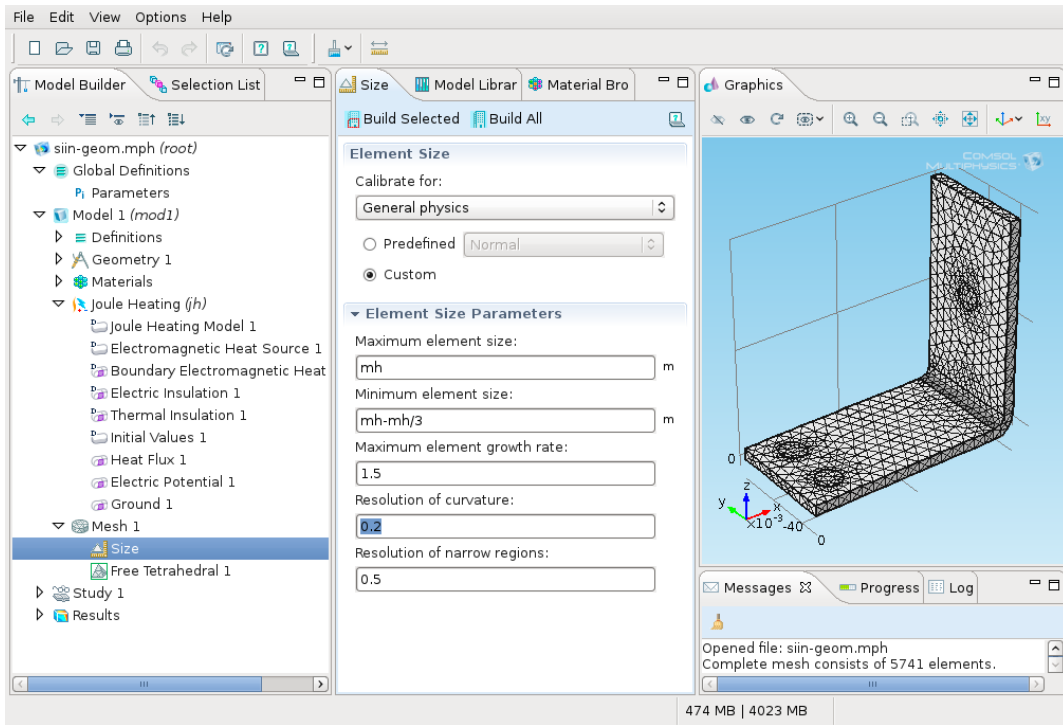


Figure 36: Changing the mesh settings.

growth rate within the domain from the smallest upto the largest. Higher value results in the higher growing speed. Value  $I$  has now influence on the growth.

- **Resolution of narrow regions** has an analogue impact on the resolution on the curvatures.

4 Click the **Build All** button to create the mesh.

---

### Task 3:

- 1 Plot the mesh of the busbar.
  - 2 What is the number of the elements?
  - 3 What is the average quality of the elements?
- 

## Simulation

To start the simulation: **Model Builder** → **Study 1: right button** → **Compute** or press the **F8** key.

The **Study** branch defines automatically the solution algorithm according to the selected physics interface and simulation type.

## Results

The default plot describes the temperature distribution in the busbar (Fig. 37). The temperature difference is less than 10 K in the device since copper and titan have large thermal conductivities. Variations in the temperature are more noticeable in the upper polt, what has to stand the double current compared to the bottom polts. The temperature in the device is higher compared to the temperature of the surrounding environment (293 K).

---

**Task 4:**

- 1 Plot the default solution.
- 2 Rotate the busbar in the rendering window to see its backside.
- 3 Restore the default view.
- 4 It is possible to alter the setting of the temperature scale manually to visualise temperature differences for different parts of the geometry. In the case of copper: **Model Builder** → **Results** → **Temperature** → **Surface 1**. Make active **Manual color range** in the **Surface** parameter window and set the **Minimum** value to **323** (Fig. 38).
- 5 Click the **Plot** button to rerender the plot.
- 6 Peek at the backside of the bushbar.
- 7 Plot the view with the changed temperature scale using **Rainbow** as a scale type.

---

**Remark:** Temperature distribution is symmetrical with respect to the mirror plane bisecting the distance between the bottom polts and the upper polt itself. In the present case the model does not need much of the computing power, so the whole geometry can be modelled. If the model is more complex, it is wise to consider the use of the symmetry to reduce the size of the model.

---

- 8 In order to visualize the current density in the busbar generate a surface plot: **Model Builder** → **Results: right button** → **3D Plot Group**. **3D Plot Group 2: right button** → **Surface** (Fig. 39).
  - 9 Click the **Replace Expression** button in the **Surface 1** parameter window and select **Joule Heating (Electric Currents)** → **Currents and charge** → **Current density norm (jh.normJ)**. *jh.normJ* is the length (absolute value) of the current density vector. Its label can be entered directly into the **Expression** field.
  - 10 Click the **Plot** button. The resulting plot is almost uniformly colored since the contact surfaces of the polts have the highest current density.
  - 11 To make the current density distribution more visible check **Manual color range** and replace the default value of the **Maximum** by **1e6**.
  - 12 Plot the current density. The resulting plot shows the shortest current path in the 90 degree bend of the busbar. At the same time the current density at the edges of the busbar is low.
-

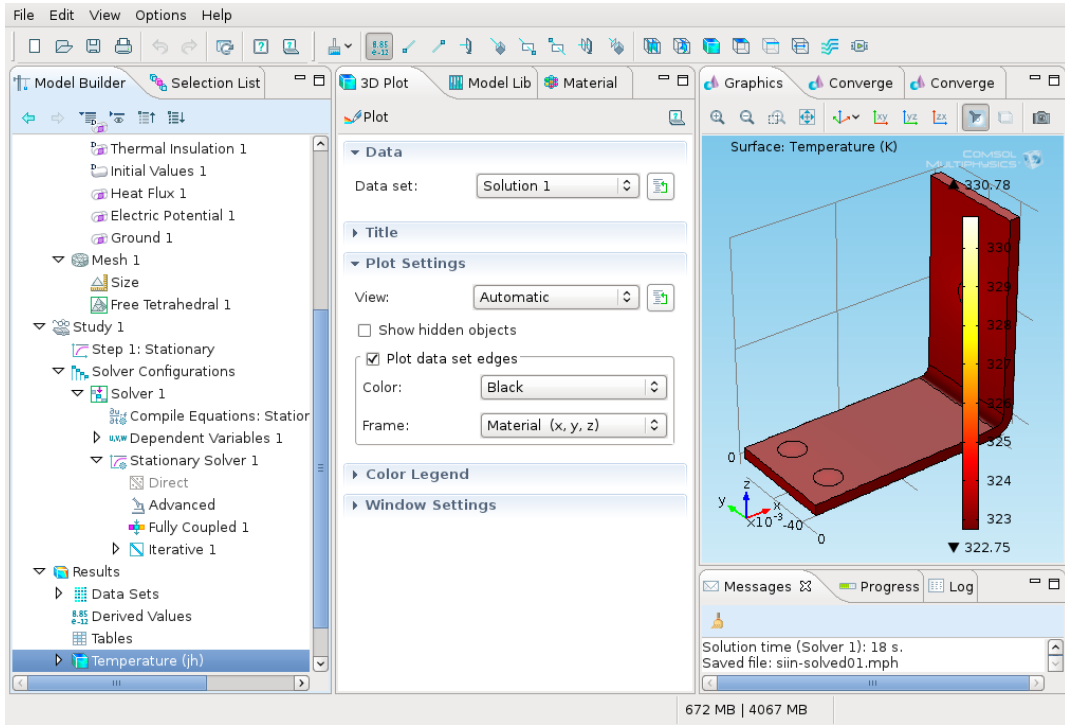


Figure 37: Default plot: the temperature.

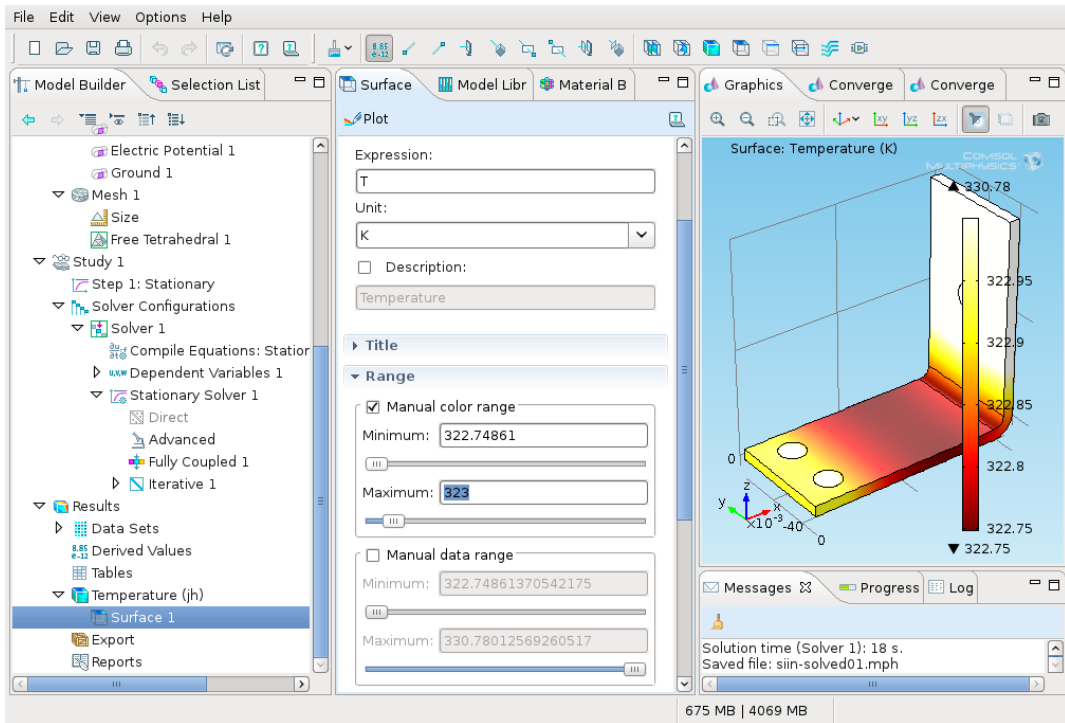


Figure 38: Changing the temperature scale.

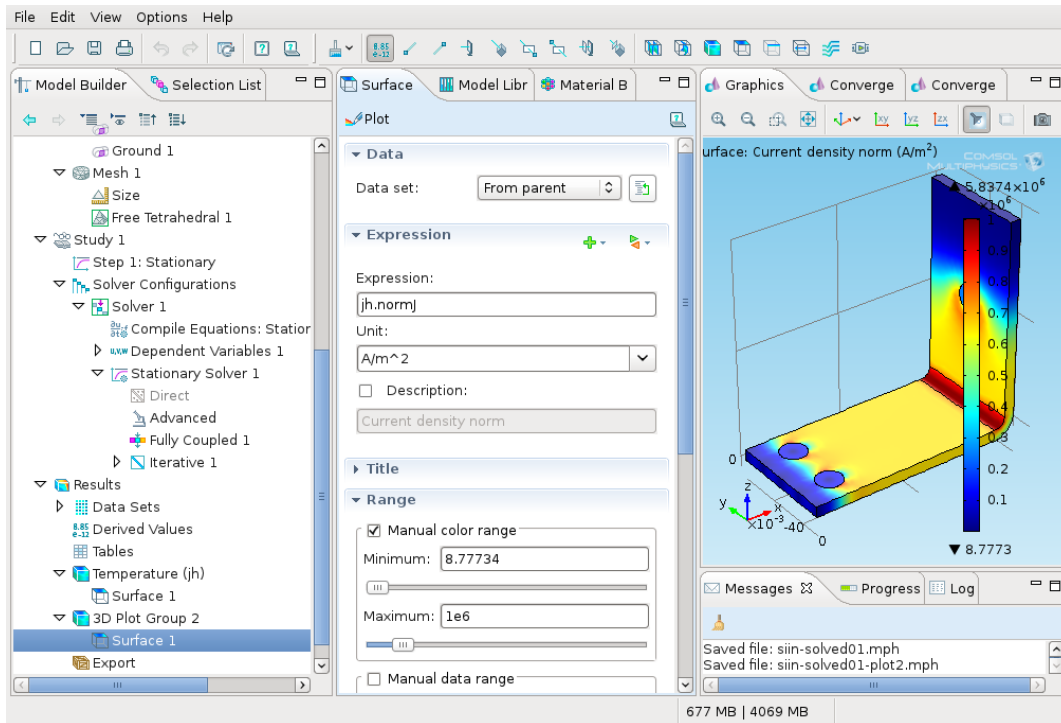


Figure 39: Current density plot.

---

**Remark:** Save the model into the *busbar-solved.mph* file. It is needed in further simulations.

---

Turn the model around and solve its backside as a thumbnail image.

## Generating the Images

Every solution may have an introductory image. To generate it select **File → Save Model Thumbnail** if the desired plot is ready.

There are two ways to generate the images:

- 1 Click the **Image Snapshot** button, what generates the snapshot of the rendering window (Fig. 40).
- 2 Add the **Image** section to the **Export** branch: **Export: right button → 3D Image** (Fig. 41). The benefit of this method is that it only needs a refresh of this branch to generate the new plot if the situation in the rendering window changes.

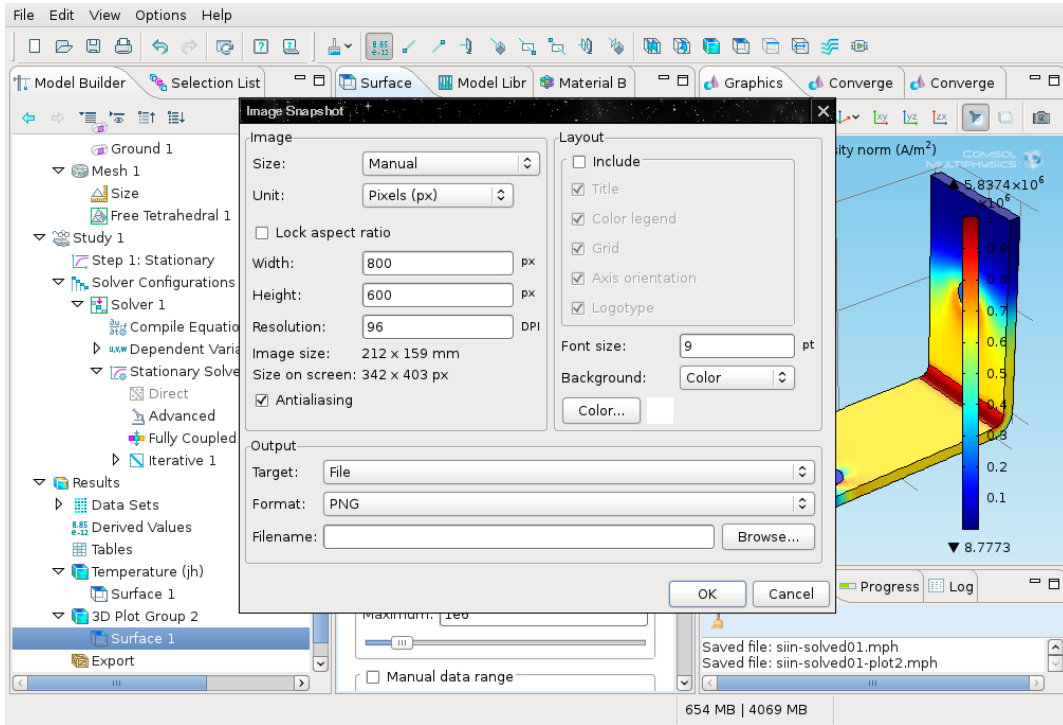


Figure 40: Saving the snapshot.

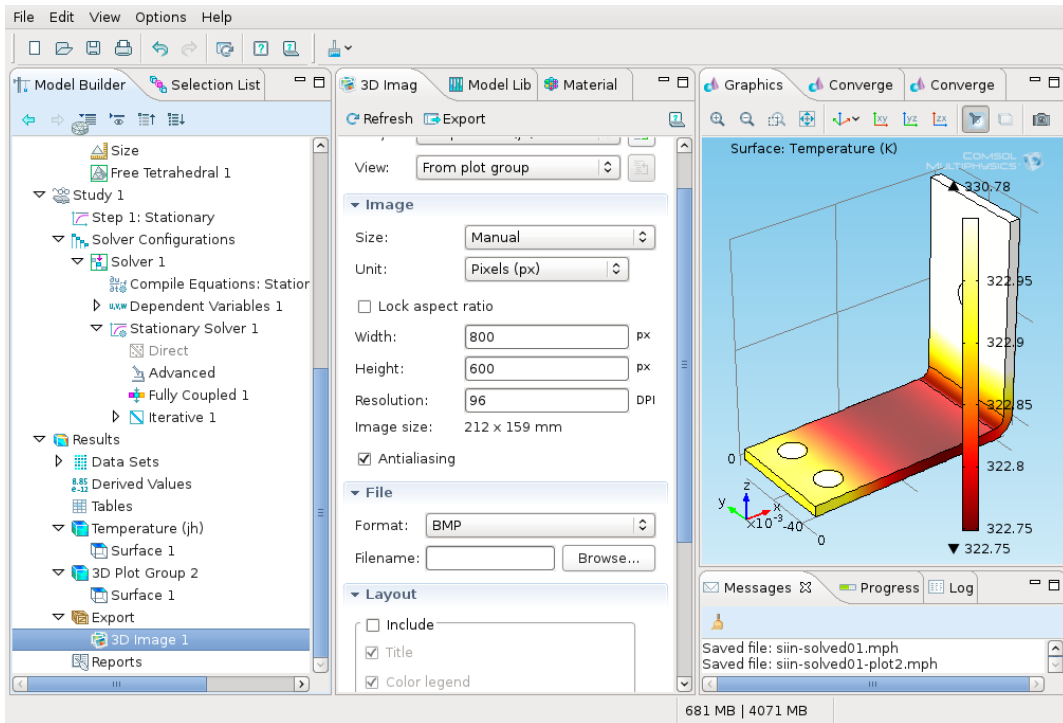


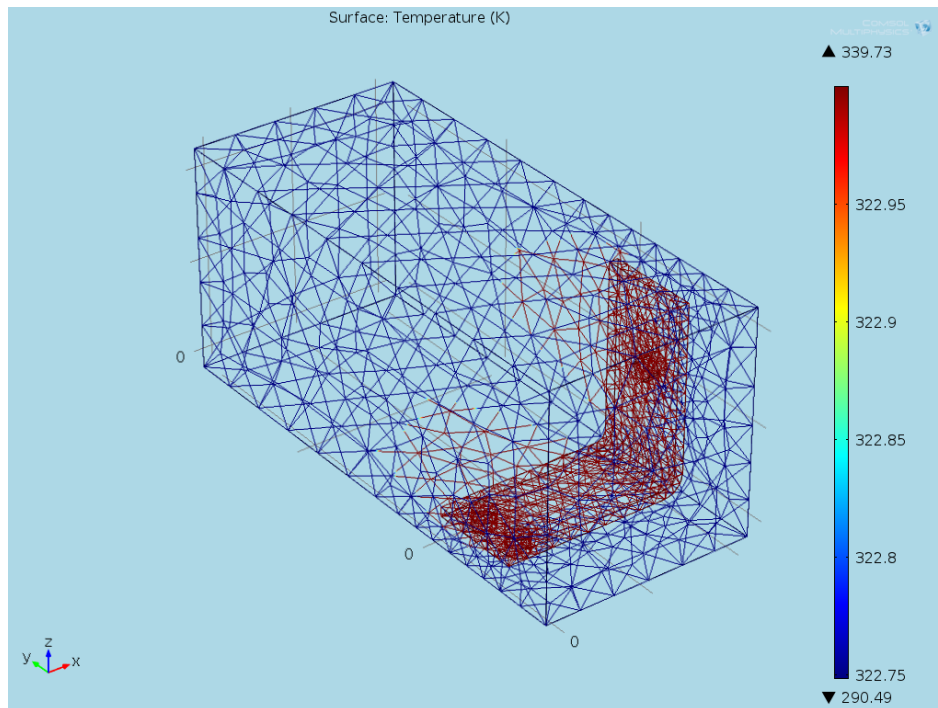
Figure 41: Exporting the plot.

# Computational Physics II

## Practical Work

### Comsol Multiphysics

#### Introduction 2



Heiki Kasemägi

October 15, 2013

# Contents

<b>Introduction</b>	<b>2</b>
Passing the Practical Work . . . . .	2
References . . . . .	3
<b>Example: Improving the Busbar Model</b>	<b>4</b>
Defining the Functions . . . . .	4
Material Properties and Databases . . . . .	6
Customising the Materials . . . . .	6
Adding Meshes . . . . .	7
Adding Structural Mechanics to the Model . . . . .	9
Study Sequence of the Joule Heating and Thermal Expansion . . . . .	11
Adding the Deformation Plot . . . . .	12
Air stream . . . . .	14
Creating the Air Box . . . . .	14
Adding the Air . . . . .	16
Adding the Air Flow . . . . .	16
Reducing the Mesh Resolution . . . . .	22
Simulation Sequences: Fluid and Joule Heating . . . . .	22
Parametric Sweep . . . . .	25
Adding the Parametric Sweep . . . . .	25
The Results of the Parametric Sweep . . . . .	28
Adding the Plots . . . . .	31



# Introduction

This practical work gives an overview of the parameters, functions, variables and model couplings. During the exercise the busbar Joule Heating model gets some improvements to cover the structural stresses and deformations due to the thermal expansion, also the influence of the cooling air stream. All surfaces, except the contact surfaces between the bolts and the busbar are cooled by the air in the form of the natural convection. The assumption is that the cross-section of the bolt does not participate in the external cooling or heating process.

*Global Definitions* and *Definitions* contain the functionality to prepare the model input and couplings and organize the simulation.

*Functions*, what is available both in the Global Definitions as well as in the *Definitions* sections, contain predefined functions useful for building the multiphysical simulation. For example, the *Step* functions makes possible to create a smoothstep function for different kind of switches.

To illustrate the use of the functions suppose that the time-dependent simulation is performed on the busbar model by applying the electric potential all over the busbar and increasing the potential value from 0 V upto 20 mV during 0.5 s. In this case, the step function can be used. The step function value is multiplied by the value of the *Vtot*

## Passing the Practical Work

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading “**Task #**”. The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may rise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size, title(s), legend(s) etc. The report and accompanying files are submitted via moodle.

## References

“Introduction to COMSOL Multiphysics” Version 4.3.a.

# Example: Improving the Busbar Model

## Defining the Functions

The present model benefits from the busbar model created and solved within the Practical Work “Introduction” and saved as *busbar-solved.mph*. The first step is to open this file.

- 1 **Global Definitions:** *right button* → *Functions* → *Step*.
- 2 In the parameters window of the *Step*: *Location* = 0.25 (Fig. 1).
- 3 *Size of the transition zone* = 0.5 specifies the width of the smoothing interval.
- 4 Click the *Plot* button. The result is in the Fig. 2.
- 5 It is possible to rename and describe the function: *Step 1: right button* → *Properties*.
- 6 Enter the desired information in the *Properties* window.

---

**Remark:** The *Global Definitions* and *Definitions* sections may contain a subsection called *Variables* holding the expressions for the dependent variables, which are initiated during the simulation. It is possible to define the global variables usable simultaneously in different models.

---

- 7 Rename *Model 1* to *busbar: Model 1*: *right button* → *Rename* → *busbar*.
- 8 Save the model into the *busbar-solved.mph* file.

---

### Task 1:

- 1 What is the purpose of the parameter *Step* of the function *Step*? Characterise it with 2 plots.
- 2 Create and plot a following function on the basis of the *Ramp* function:

$$y(x) = \begin{cases} 0, & x < 0 \\ 1, & x > 0.2 \end{cases}$$

---

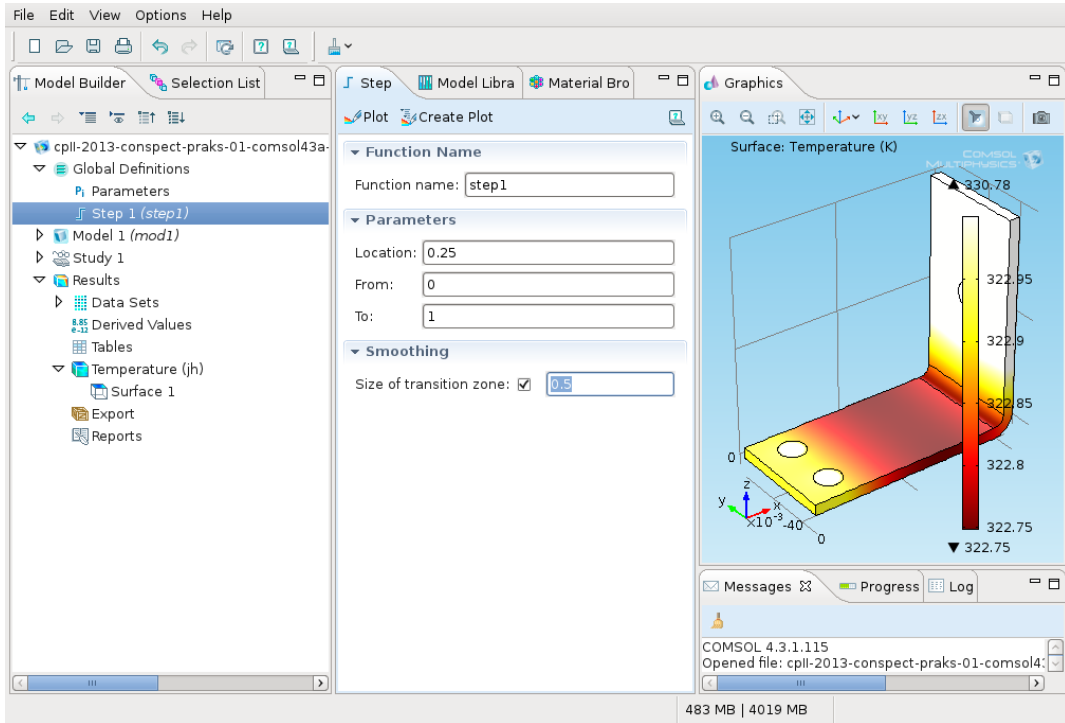


Figure 1: Creating the step function.

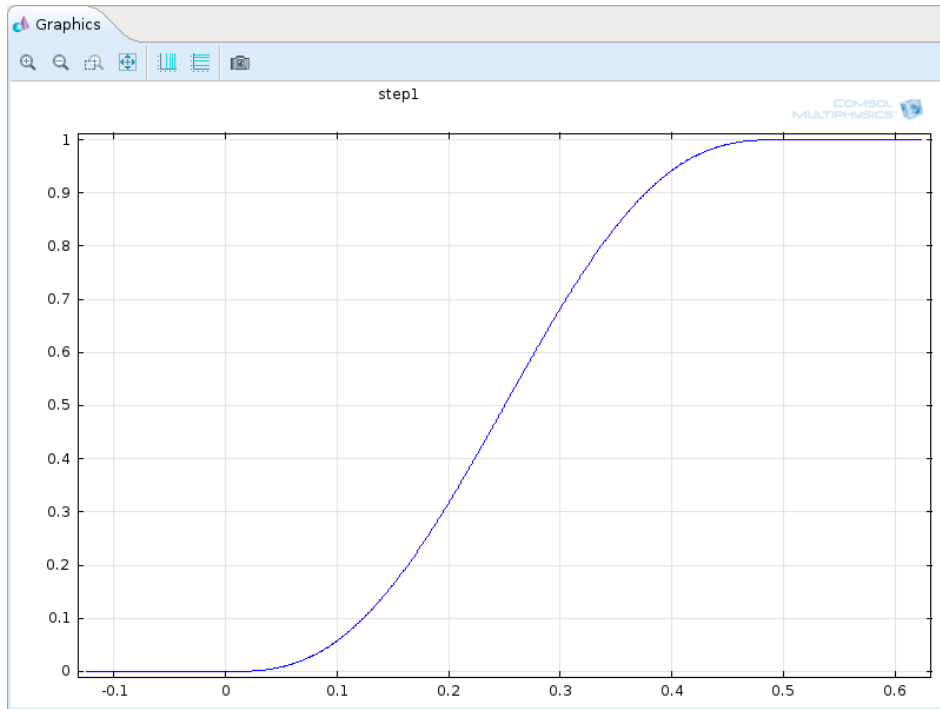


Figure 2: The step function

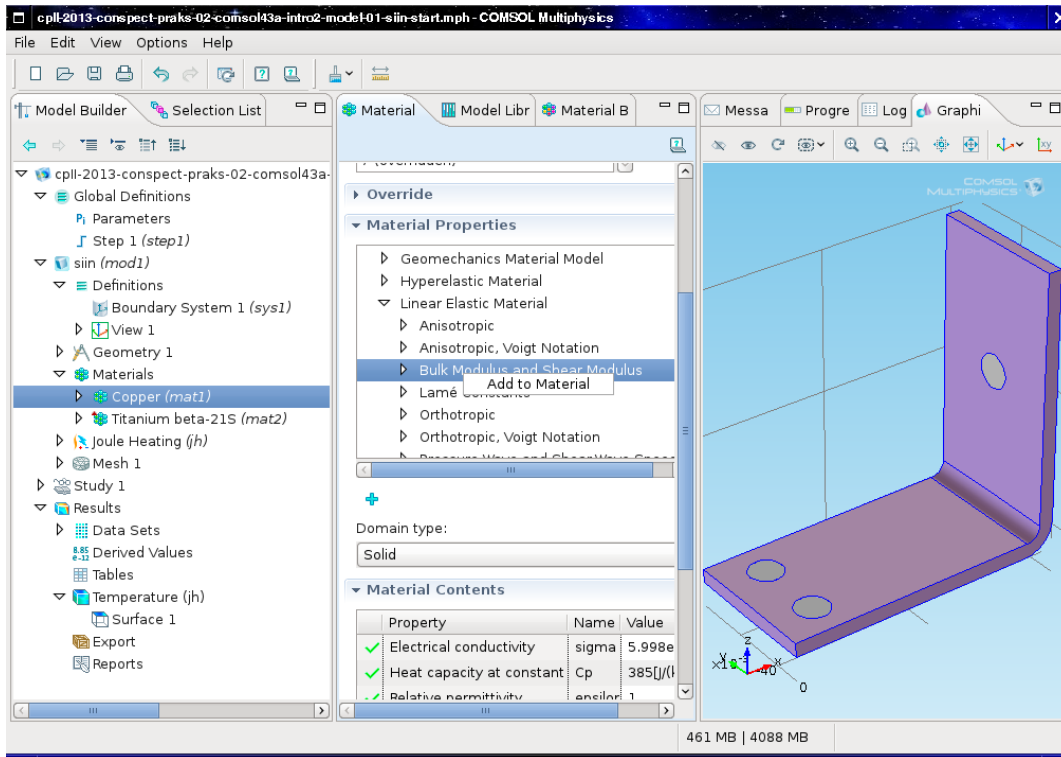


Figure 3: Adding a material property.

## Material Properties and Databases

So far the functionality of the *Materials* section was used for the properties of titan and copper in the model. This section makes possible to define custom material properties and create a custom material database. Also the properties can be added to the existing materials. If the property is dependent on any variable, it is possible to generate the plot of the dependence to make sure if it is correct. Typically, it is the temperature-dependence but not only.

*Material Library* plug-in contains about 2500 material descriptions with tens of thousands of properties depending on the temperature.

## Customising the Materials

This section describes how to add a property to the existing material. These properties are the *bulk modulus* and the *shear modulus* for copper.

1 *Model Builder* → *Materials* → *Copper*.

2 *Material Properties* table contains the list of all definable properties. To add the bulk modulus and the shear modulus for copper: *Solid Mechanics* → *Linear Elastic Material* → *Bulk Modulus and Shear Modulus*: right button → *Add to Material* (Fig. 3).

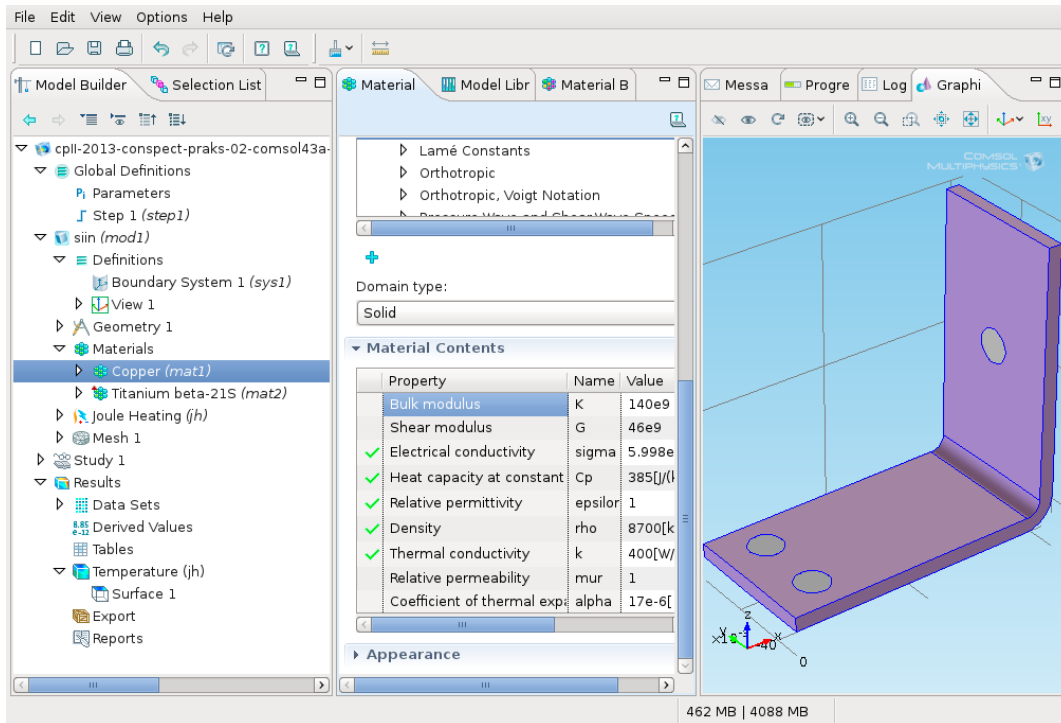


Figure 4: Values of Material Properties.

- Bulk modulus** ja **Shear modulus** appear now in the *Material Contents* table. The exclamation mark as a warning sign shows that the values are not defined. Enter **140e9** into the **Value** column for the **Bulk modulus** and **46e9** for the **Shear modulus** into the same column (Fig. 4).
- Now the properties of the copper are altered. But these can not be saved into the **Solid Mechanics** material database since it has a write protection. It is possible to save them into the custom material database: **Model Builder** → **Copper: right button** → **Add Material to Library** → “**User Defined Library**”.

## Adding Meshes

The model may contain the meshing sequences with different parameters. These sequences can be available for all simulation steps. Each simulation can decide which particular mesh it wants to use. In the busbar model, the second mesh is added with higher resolution in the areas around the polts and the bend.

- Save the current model into the separate file **busbar\_1.mph**.
- Add the second mesh: **busbar(mod1): right button** → **Mesh**. All meshes reside now in the **Mesh** branch.

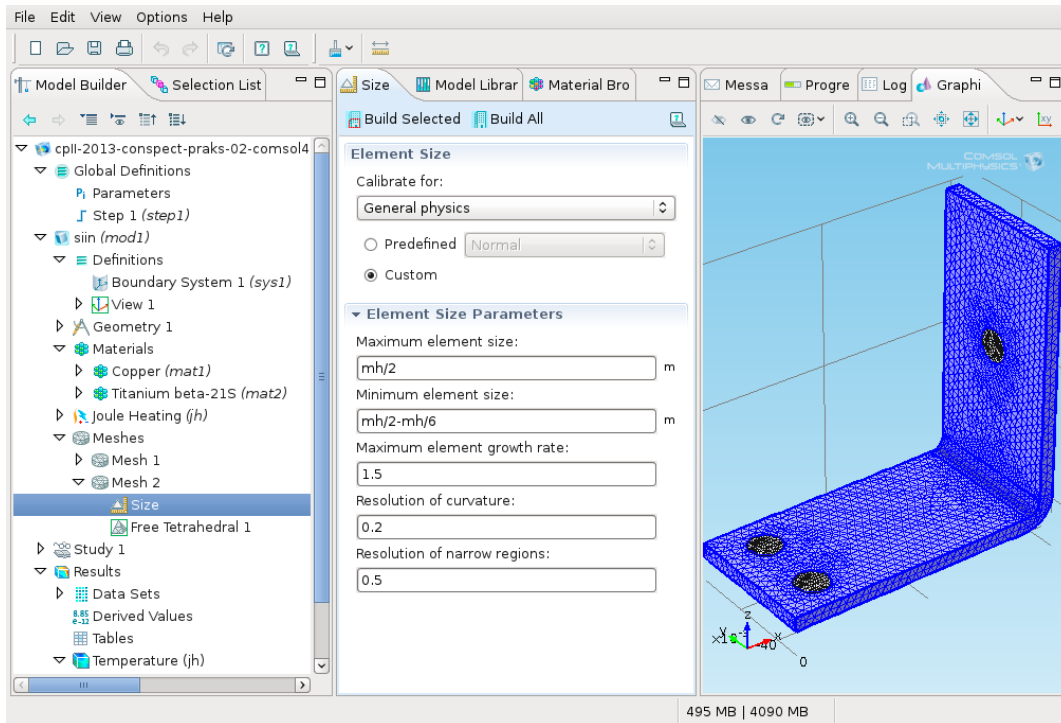


Figure 5: The parameters of the new mesh.

---

**Remark:** *Model 1* was renamed *busbar* in the previous step. To select an optimum combination of names, tags and identifiers: *View* → *Model Builder* → *Node Label*.

---

**3** *Mesh 2* → *Mesh Settings* → *Sequence type* = *User-controlled*.

**4** *Model Builder* → *Mesh 2* → *Size* → *Element size* → *Custom* (Fig. 5):

- *Maximum element size* =  $mh/2$
- *Minimum element size* =  $mh/2 - mh/6$
- *Resolution of curvature* =  $0.2$

**5** Click the *Build All* button.

Clicking the *Mesh 1* and *Mesh 2* nodes helps to compare the meshes. The alternative of using a number of different meshes is to run a parametric sweep of the maximum mesh size *mh*.

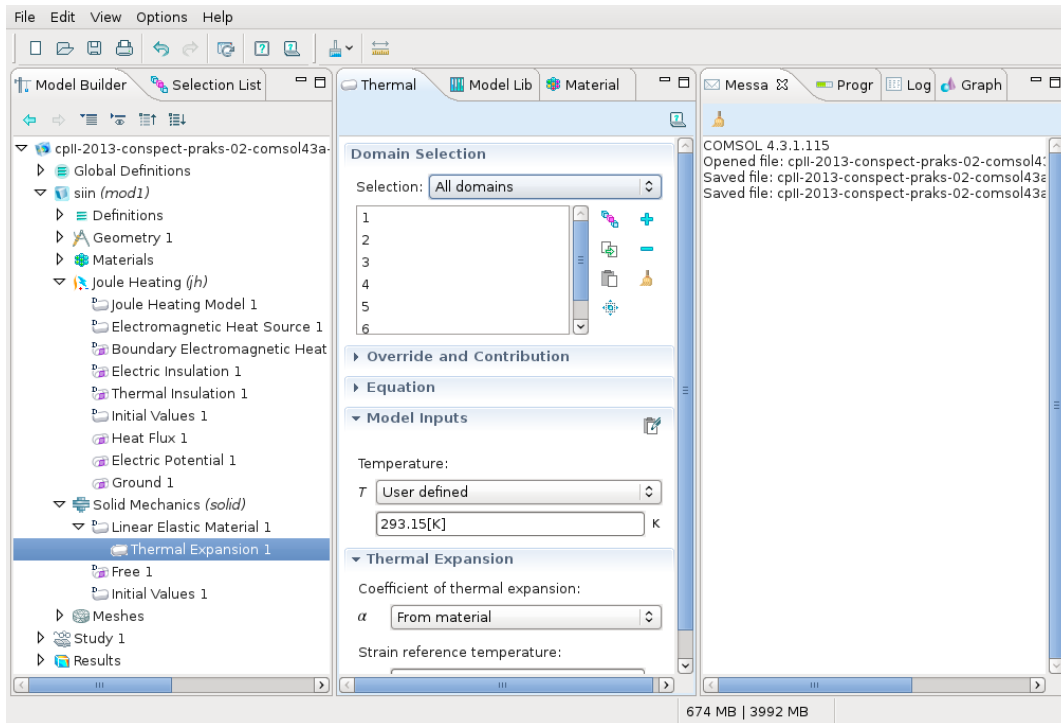


Figure 6: Adding the thermal expansion.

---

### Task 2:

- 1 Plot the mesh of the *Mesh 2*.
  - 2 Compare statistics of the both meshes to each other: the number of the elements, minimum and average quality of the elements, aspect ratio and make the conclusions.
  - 3 Which volume elements have better average quality? What might be the reason on this?
- 

## Adding Structural Mechanics to the Model

After the simulation of the Joule Heating it is clear that the temperature rises in the busbar. The next logical question would be, what is the mechanical stress due to the thermal expansion. This question can be answered by adding the Structural Mechanics interface to the model.

- 1 Open the *busbar-solved.mph* file. Save it as *busbar\_II.mph*.
- 2 *Model Builder* → *busbar*: right button → *Add Physics*.
- 3 *Model Wizard* → *Structural Mechanics* → *Solid Mechanics (solid)*: right button → *Add Selected*.



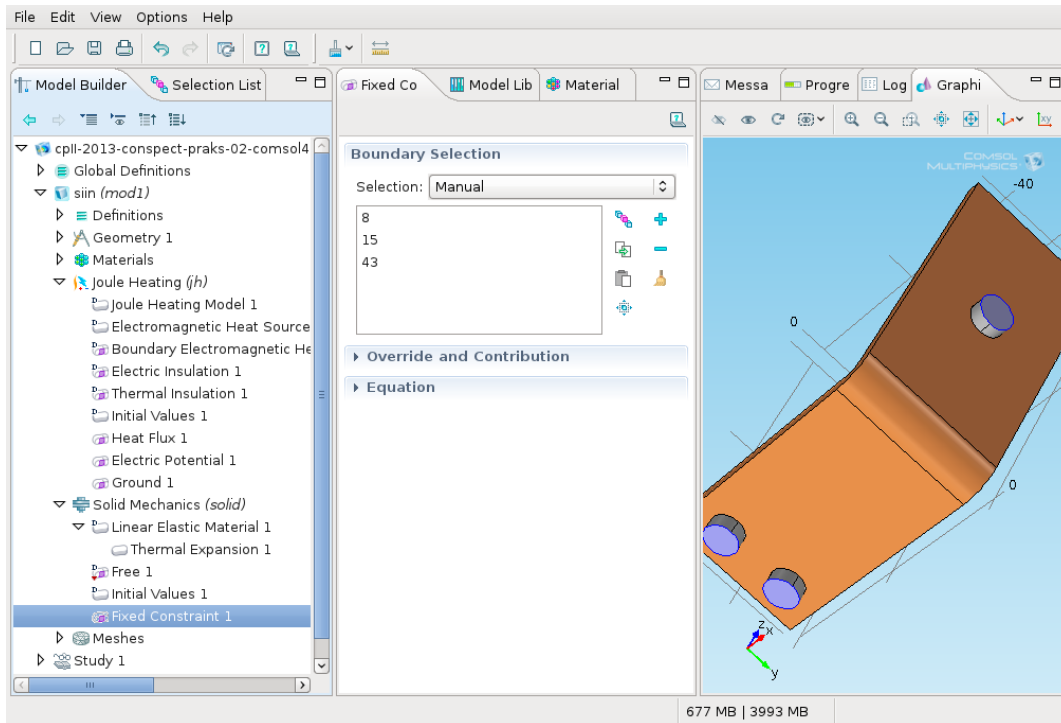


Figure 7: Constraining the busbar.

4 Click the **Finish** button and save the file.

---

**Remark:** While adding the new physics interface, one has to ensure that the materials in the **Materials** section have all properties what the new interface requires. In this case copper and titan have all necessary properties present.

---

5 Now it's time to add the thermal expansion: **Model Builder** → **Solid Mechanics** → **Linear Elastic Material 1: right button (domain level)** → **Thermal Expansion** (Fig. 6).

6 **Model inputs** → **Temperatur = jh/jhml (select from the list)**. This is the temperature field from the Joule Heating coupling the Joule Heating to the thermal expansion of the busbar.

---

**Remark:** The Structural Mechanic interface has predefined sections for the thermal stress and deformation.

---

7 Next constrain the busbar at the titan polt: **Model Builder** → **Solid Mechanics: right button (boundary level)** → **Fixed Constraint**.

8 Click the **Fixed Constraint 1** branch. Turn the back side of the busbar forth in the rendering window. Add the boundaries **8**, **15** and **43** into the selection list (Fig. 7).

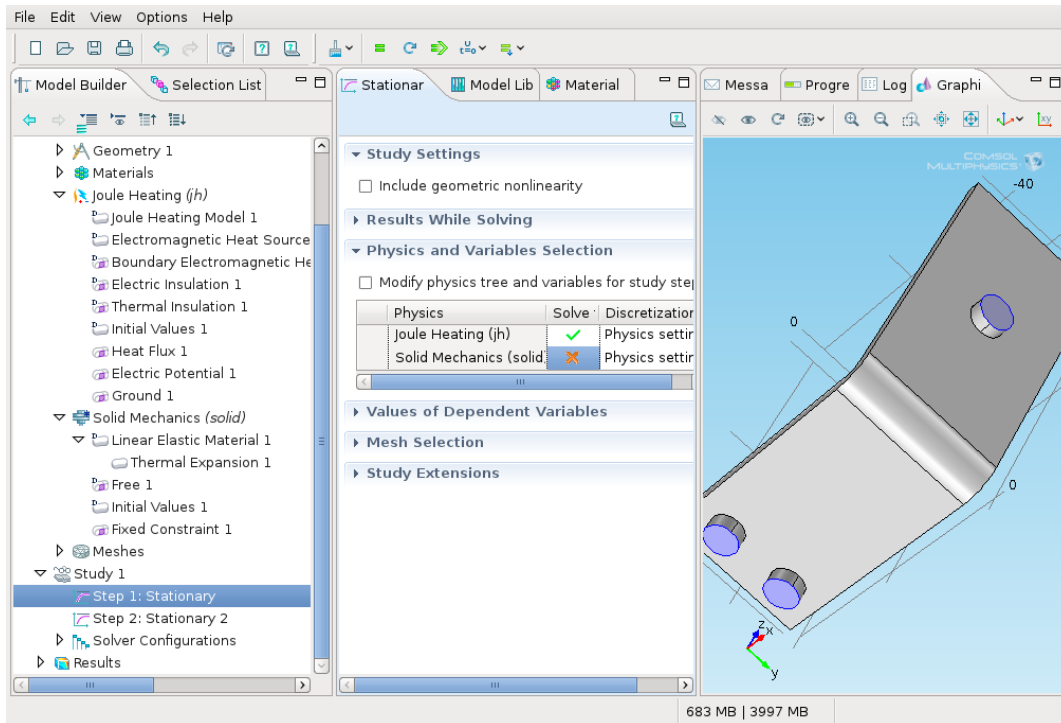


Figure 8: Removing the Solid State Mechanics from the first simulation step.

## Study Sequence of the Joule Heating and Thermal Expansion

The Joule Heating is independent of the stresses and strains in the busbar, assuming that the deformations are small and the electric contact pressure can be ignored. This means that the model can be run with the temperature as an input to the structural analysis. In other words, the extended multiphysical problem is weakly coupled. In this case, the model can be solved in two steps - starting with the strongly coupled Joule Heating following by the structural analysis.

- 1 Adding the second step: *Model Builder* → *Study 1: right button* → *Study Steps* → *Stationary*.

---

**Remark:** While adding the new simulation steps, the manual corrections have to be done to connect the correct physic interface to the correct simulation step. In the present case the structural analysis needs to be removed from the first simulation step.

---

- 2 *Study 1* → *Step 1: Stationary*.

- 3 In the *Physics And Variables Selection* table click in the *Solve* cell of the *Solid Mechanics (solid)* row to turn the green checkmark into the red cross marking the removal of the *Solid Mechanics* from the first step (Fig. 8).

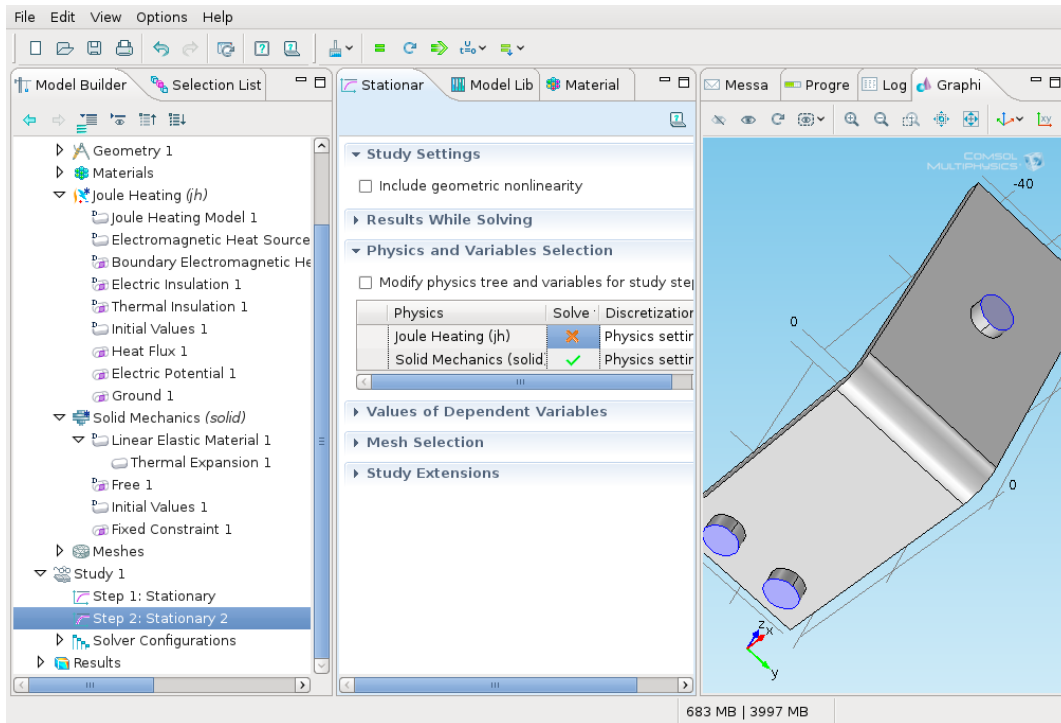


Figure 9: Removing the Joule Heating from the second simulation step.

4 By repeating these steps, remove the Joule Heating from the second step: *Study 1* → *Step 2: Stationary 2* → *Physics and Variables Selection* → *Joule Heating*: click the *Solve* cell (Fig. 9).

5 Solve the problem: *Study 1*: right button → *Compute* or press the *F8* key.

Save the file after the calculations as *busbar\_II.mph*.

---

**Task 3:** Plot the default solution.

---

## Adding the Deformation Plot

Another plot group is added to the *Results* in order to plot the deformations.

1 *Results*: right button → *3D Plot Group* → *3D Plot Group 2*: right button → *Surface* (Fig. 10).

2 Click the *Surface 1* branch. *Expression* → *Replace Expression* → *Solid Mechanics* → *Displacement* → *Total Displacement*. The another way is to enter *solid.disp* into the *Expression* field.

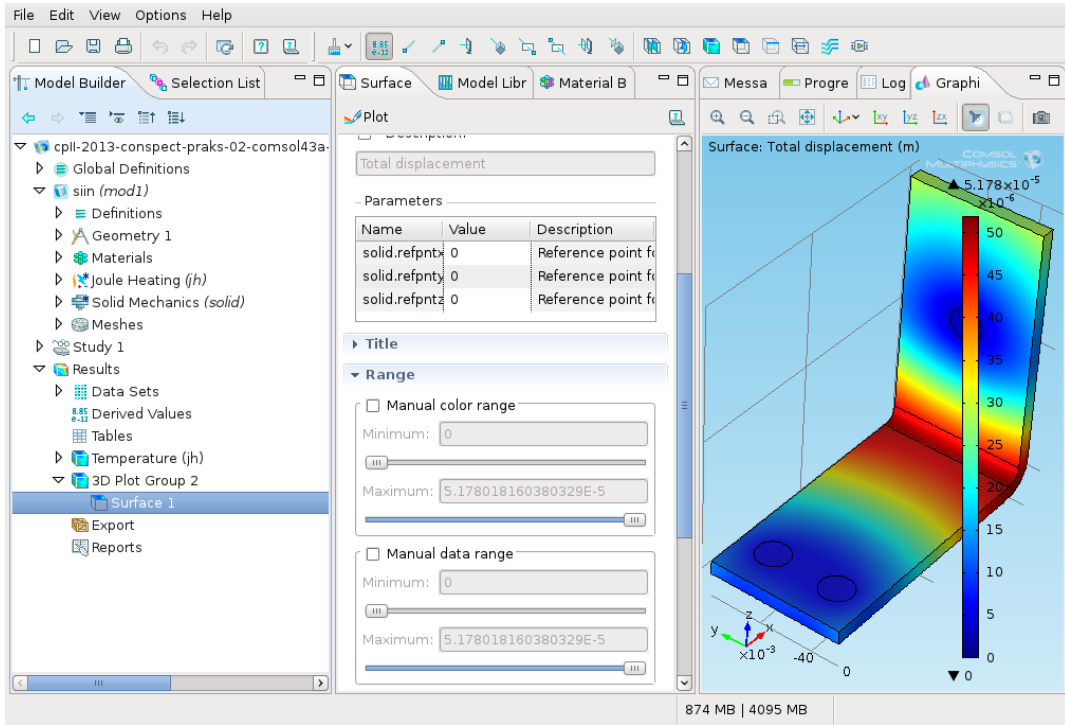


Figure 10: Adding the deformation plot.

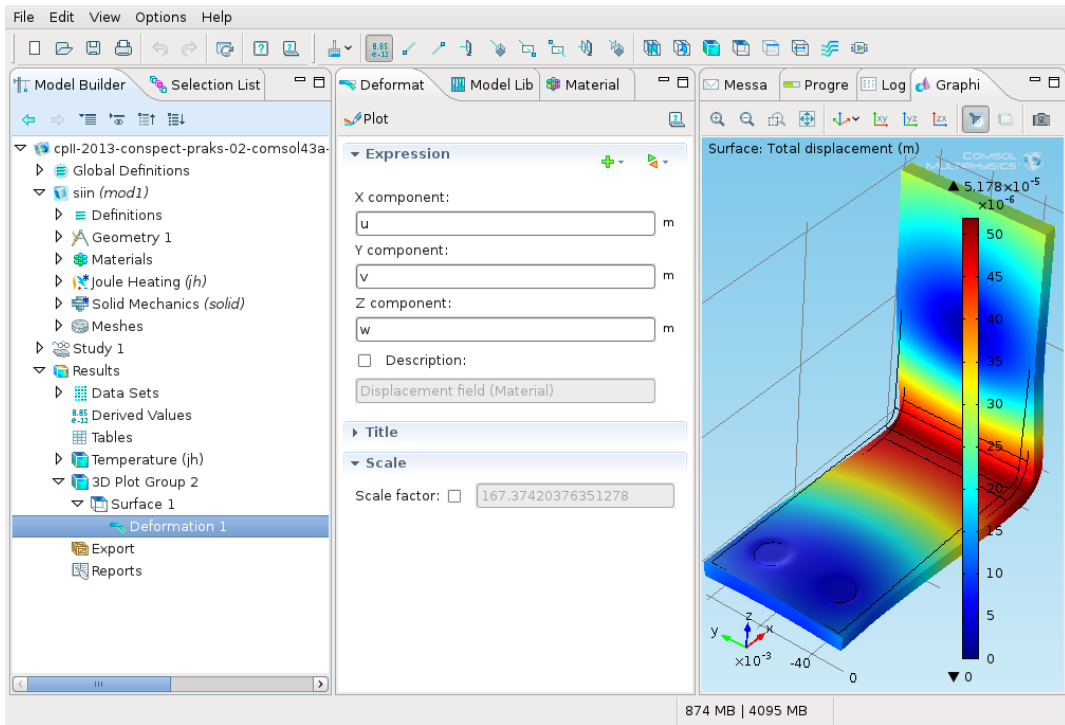


Figure 11: Deformed busbar.

- 3 **Range** → **Manual color range (uncheck)**. The total displacement due to the thermal expansion is plotted as a surface plot in the rendering window.
- 4 The next thing to add is the displacement from the initial boundaries of the busbar: **Model Builder** → **Results** → **3D Plot Group 2** → **Surface 1: right button** → **Deformation** (Fig. 11).
- 5 Save the model into the **busbar\_II.mph** file.

It is possible to plot the von Misses stress and the other quantities in the similar way.

---

#### Task 4:

- 1 Plot the deformations.
  - 2 Plot the electric current power losses in the busbar. Which area has the greatest losses?
- 

## Air stream

After the studying of the heat generated in the busbar and possible thermal stress, there may rise the question how to cool the busbar. With external air stream over the surfaces, perhaps?

By adding the air stream to the Joule Heating a new multiphysical coupling is formed. To simulate the flow domain, an airbox has to be drawn around the busbar. For that the original geometry needs a modification. The air flow has to be simulated according to the Fig. 12.

### Creating the Air Box

- 1 Open the **busbar\_I.mph** file.
- 2 **Model Builder** → **Geometry 1: right button** → **Block**.
- 3 The parameters of the **Block 1** (Fig. 13):
  - **Width** =  $L+5*tbb$ ;
  - **Depth** =  $3*L$ ;
  - **Height** =  $L+6*tbb$ ;
  - **Position: x** =  $-2*tbb$ ;
  - **Position: y** =  $-L$ ;
  - **Position: z** =  $-tbb$ ;
  - **Axis type: Cartesian**;
- 4 Click the **Build All** button.

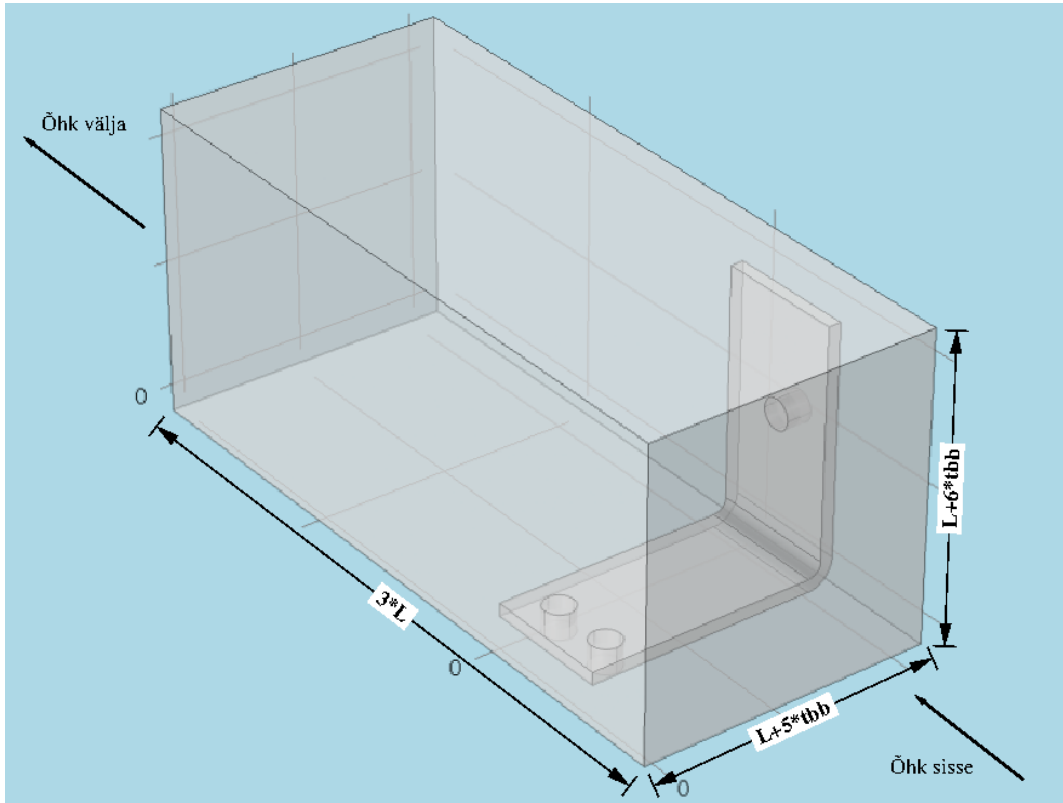


Figure 12: Geometry of the air box.

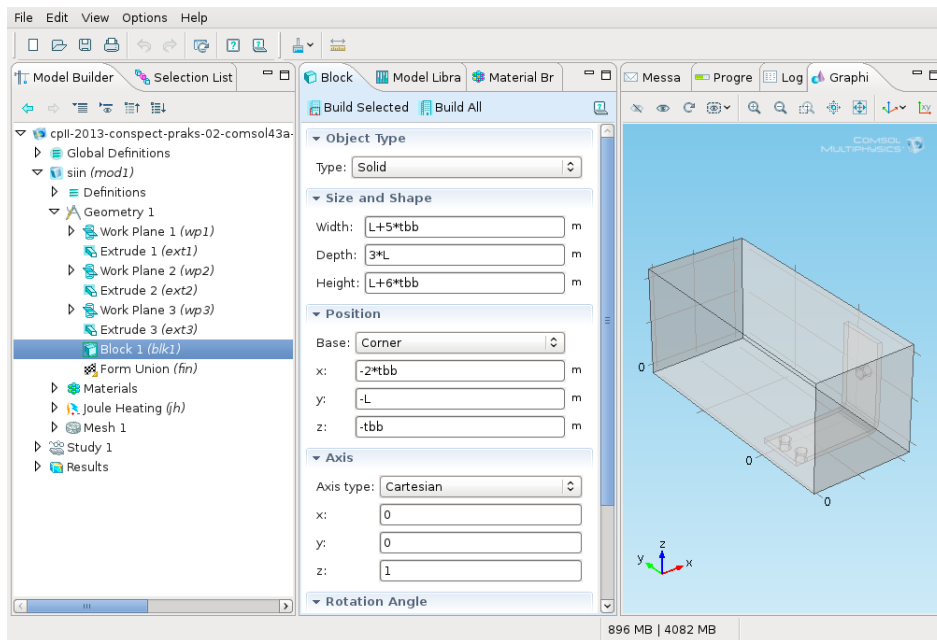


Figure 13: Adding the air box.

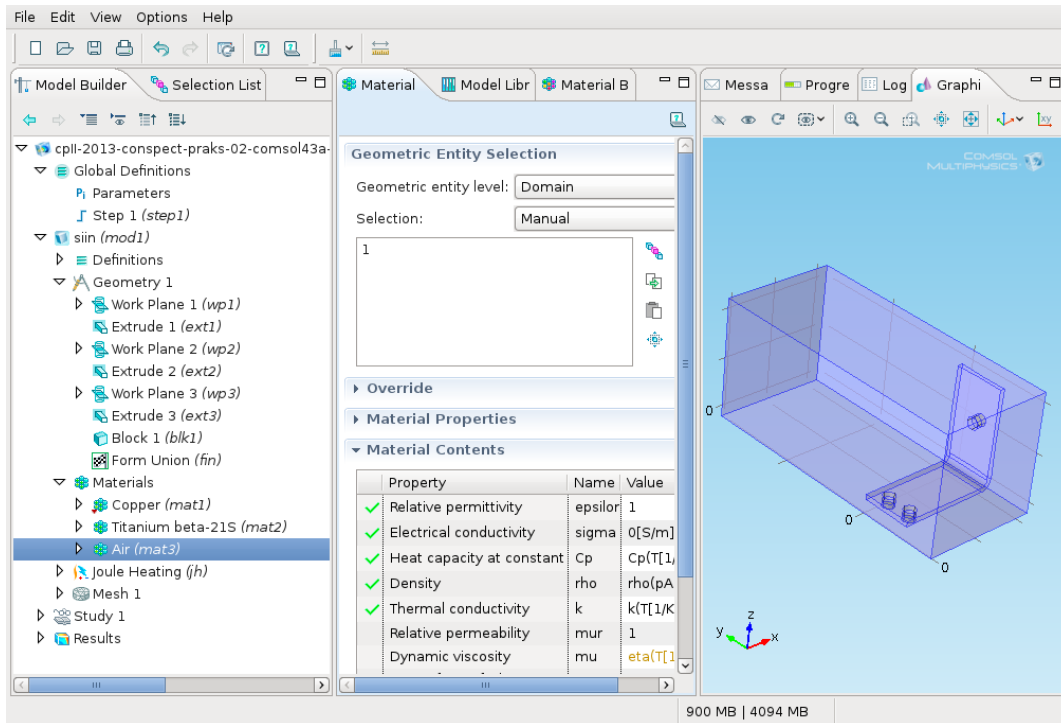


Figure 14: Adding the air.

- 5 Add a new constant  $V_{in}$  with the value of  $1e-1[m/s]$  into the *Global Definitions*  $\rightarrow$  *Parameters* table as an air flow input speed.
- 6 Save the file as *busbar-box\_1.mph*.

## Adding the Air

- 1 *View*  $\rightarrow$  *Material Browser*  $\rightarrow$  *Built-In*  $\rightarrow$  *Air*: right button  $\rightarrow$  *Add Material to Model*.
- 2 Click the *Model Builder*  $\rightarrow$  *Materials*  $\rightarrow$  *Air* branch (Fig. 14).
- 3 Click the air box in the rendering window to turn it red and add the box into the selection list.

## Adding the Air Flow

- 1 *Model Builder*  $\rightarrow$  *busbar*: right button  $\rightarrow$  *Add Physics*  $\rightarrow$  *Fluid Flow*  $\rightarrow$  *Single-Phase Flow*  $\rightarrow$  *Laminar Flow*: right button  $\rightarrow$  *Add Selected*. Click the *Finish* button.
- 2 Click the *Select Boundaries* and then *Wireframe* buttons in the toolbar of the rendering window (Fig. 15).
- 3 *Model Builder*  $\rightarrow$  *Joule Heating*: right button  $\rightarrow$  *first section (domain level)*  $\rightarrow$  *Heat Transfer*  $\rightarrow$  *Heat Transfer in Fluids*.

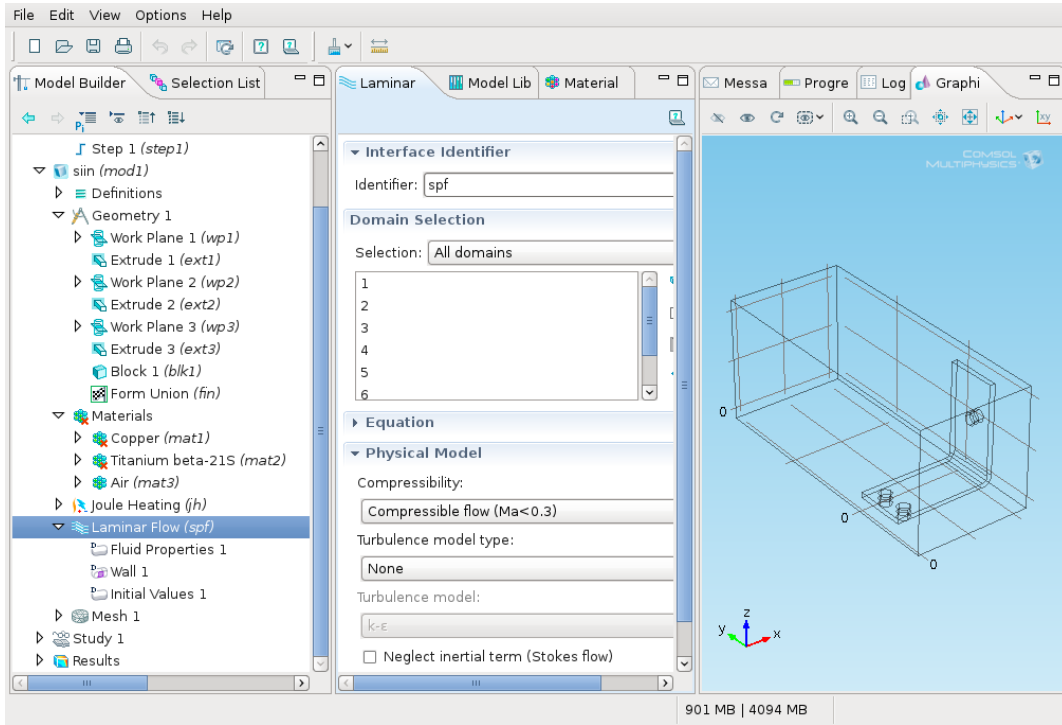


Figure 15: Adding the air flow.

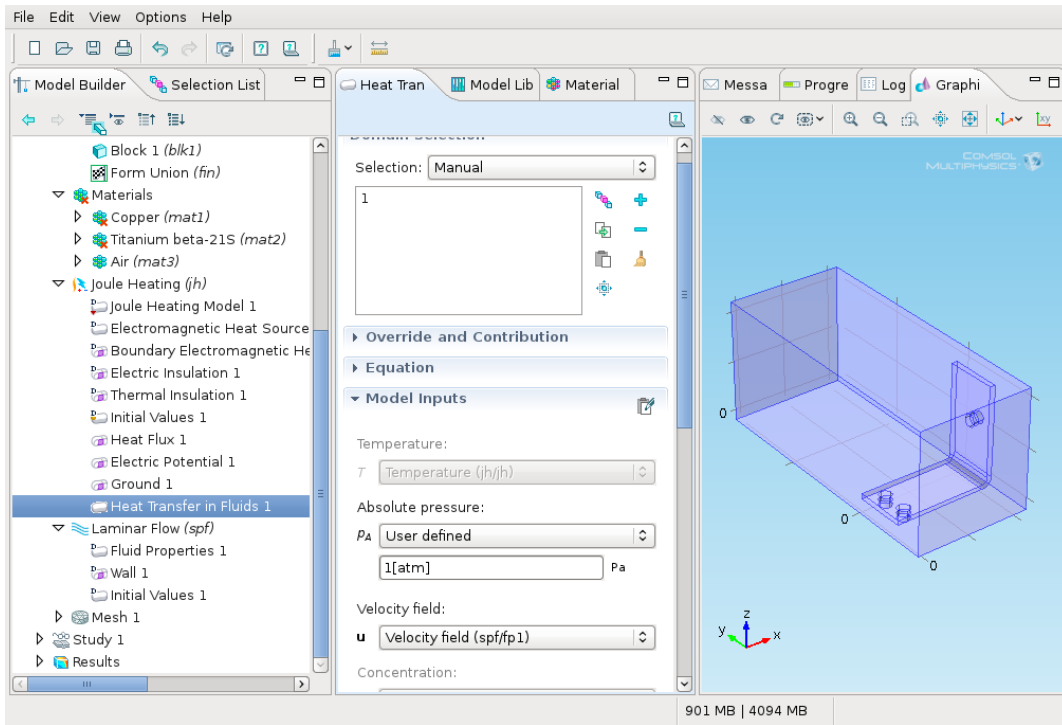


Figure 16: Thermal conductivity in fluids.



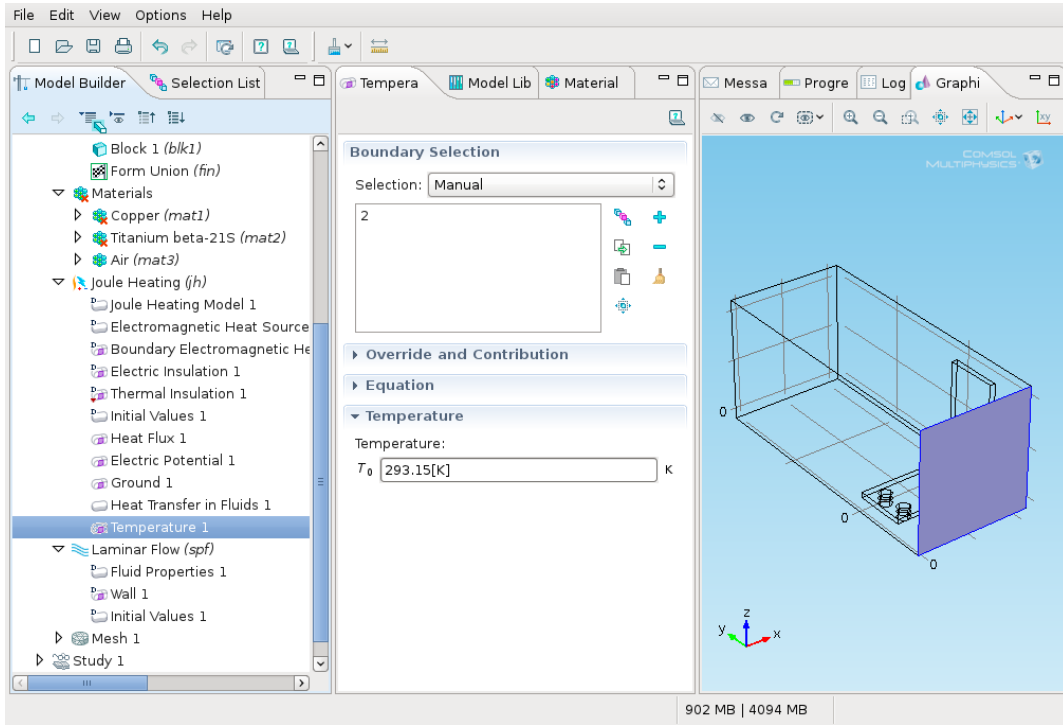


Figure 17: Temperature input boundary condition of the air flow.

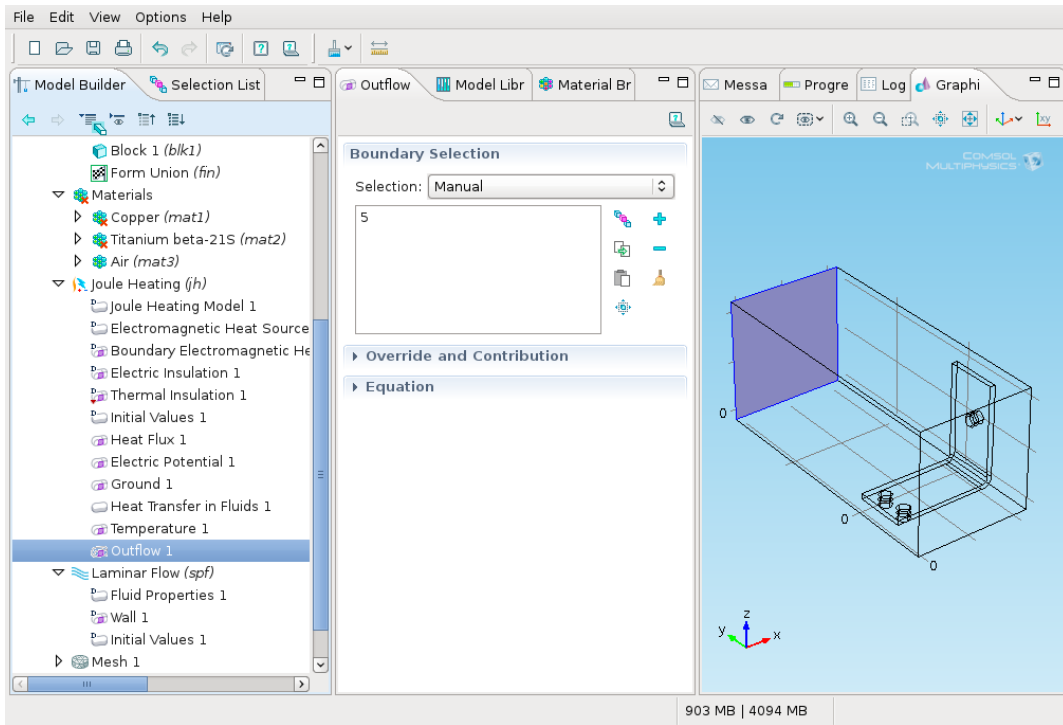


Figure 18: Outflow boundary condition of the air flow.

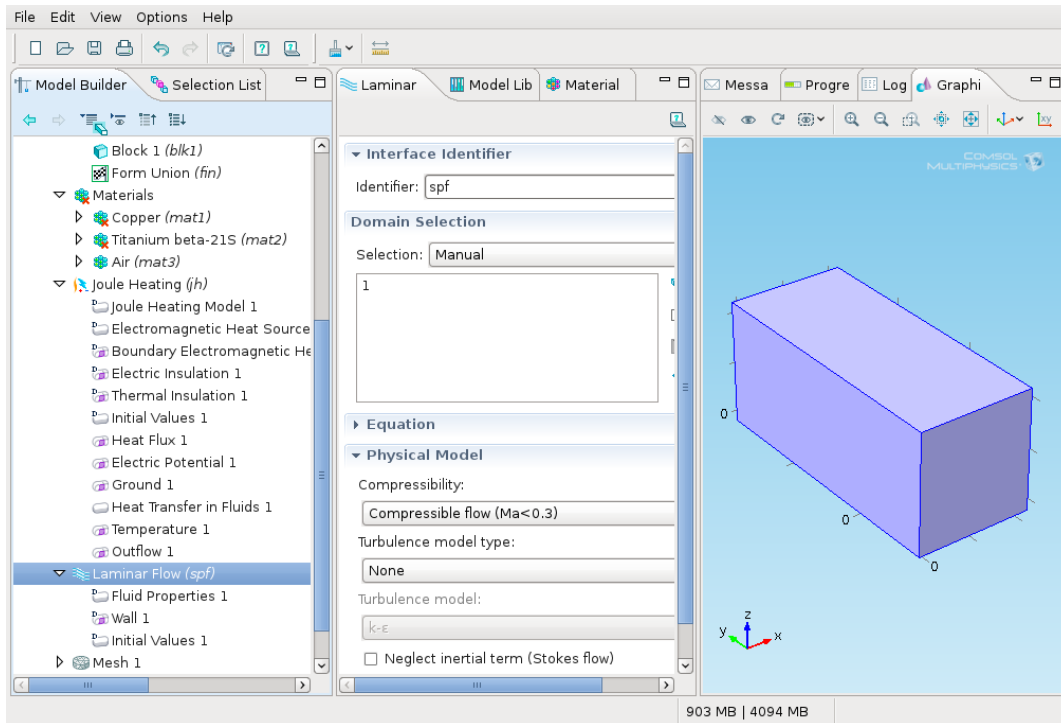


Figure 19: Flow domain.

- 4 Add the air box (domain 1) into the selection list of the *Heat Transfer in Fluids 1*.
- 5 Next the air flow and heat transfer will be coupled.

*Heat Transfer in Fluids 1* → *Model Inputs* → *Velocity field* → *Velocity field (spffp1)* (from the selection list) (Fig. 16). This couples the flow field of the *Laminar Flow* with thermal conductivity.

- 6 The thermal conductivity boundary conditions in the flow domain: *Model Builder* → *Joule Heating: right button* → *second, boundary section* → *Heat Transfer* → *Temperature*.
- 7 Add boundary 2 into the temperature selection list (Fig. 17). Click the *Transparency* button to switch the transparency off.
- 8 *Model Builder* → *Joule Heating: right button* → *boundary level* → *Heat Transfer* → *Outflow*.
- 9 Add boundary 5 into the outflow selection list (Fig. 18).
- 10 The boundary conditions for the busbar, polts, *Electric Potential 1* and *Ground 1* should remain the same.

The air flow should be active only in the flow domain and specify its input, output and symmetry conditions.

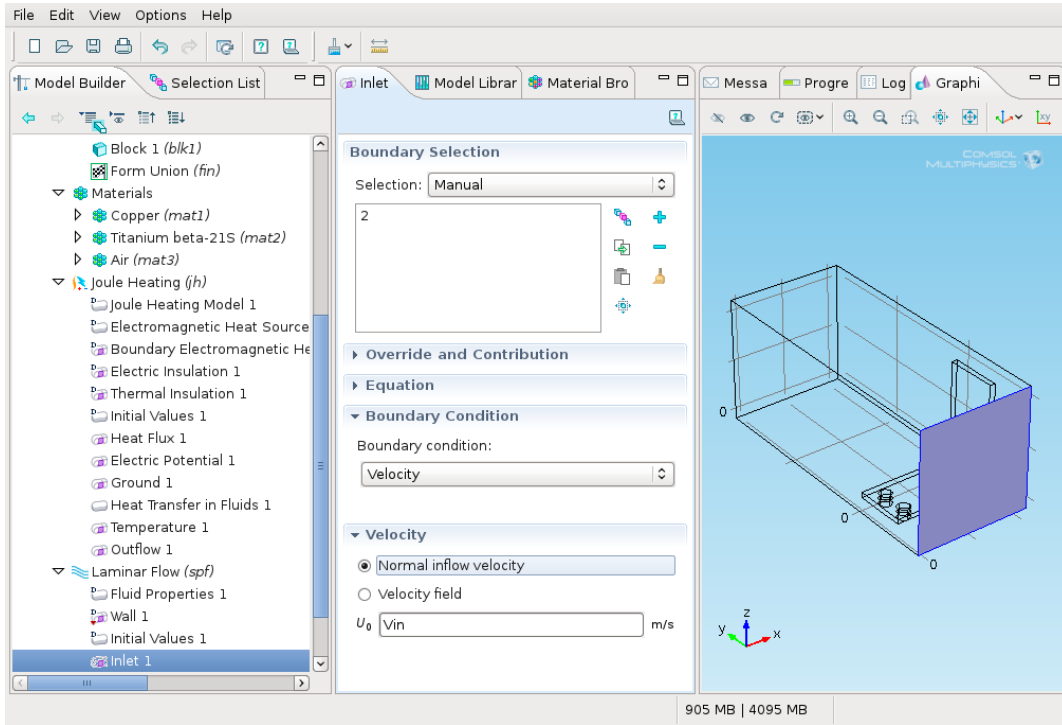


Figure 20: Air flow input boundary condition.

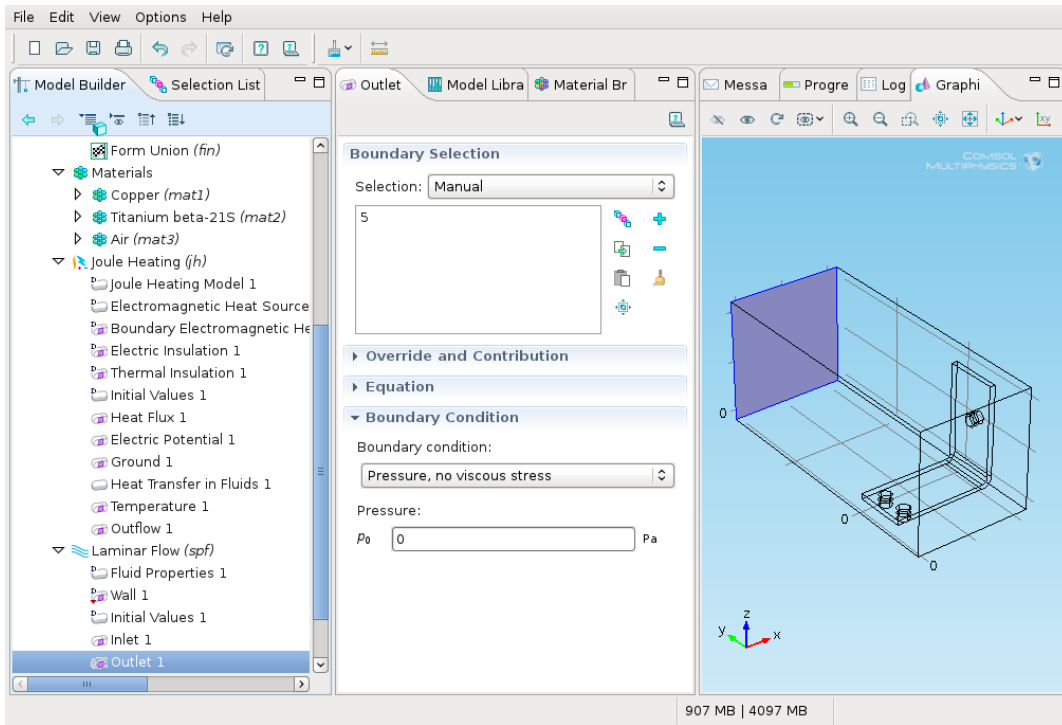


Figure 21: Air flow output boundary condition.

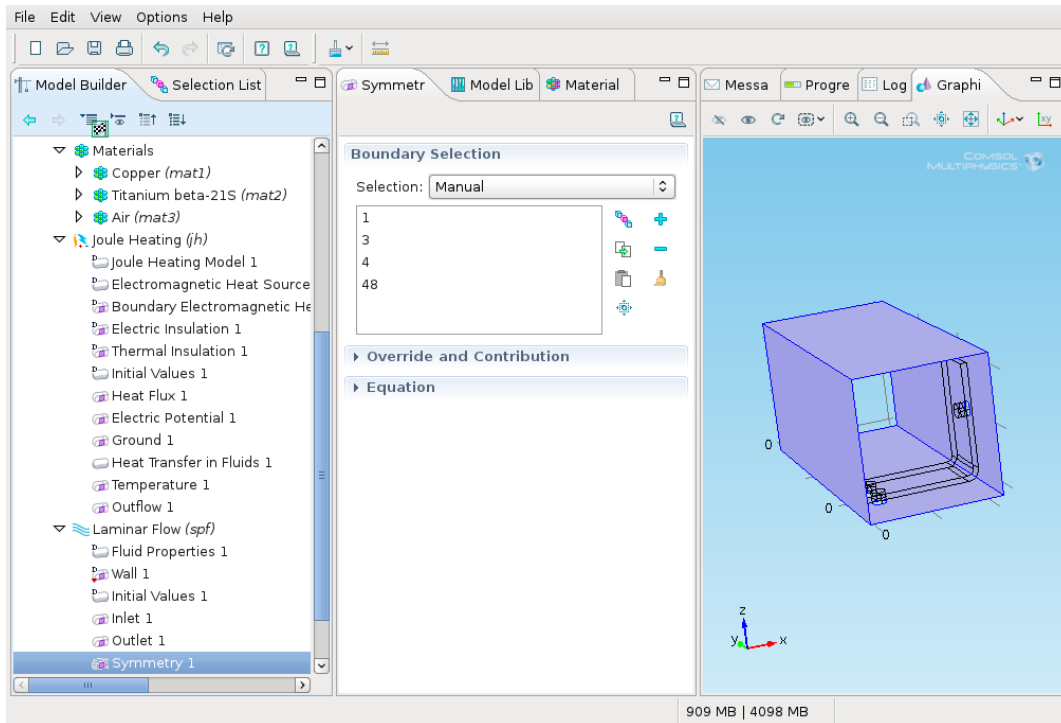


Figure 22: Symmetry condition.

1 **Model Builder** → **Laminar Flow** → **Clear Selection**.

2 Add the air box (domain *I*) into the selection list (Fig. 19).

---

**Remark:** A good practice is to check if the air has all the required material properties for the multiphysical couplings. There should be no yellow warning signs in the **Material Contents** table.

---

3 **Model Builder** → **Laminar Flow**: **right button** → **boundary level** → **Inlet** (Fig. 20).

4 Add the boundary 2 into the selection list.

5  $U_0 = V_{in}$ .

6 **Laminar Flow**: **right button** → **boundary level** → **Outlet**. Add the boundary 5 into the output selection list (Fig. 21).

7 The last step is to add the symmetry to the boundaries. It can assume that the flow just outside the channel faces is the same as outside those faces. This is expressed by the symmetry condition: (Fig. 22): **Laminar Flow** → **Symmetry**. Add the boundaries 1, 3, 4 and 48 into the selection list.

8 Save the model into the **busbar-box\_1.mph** file.

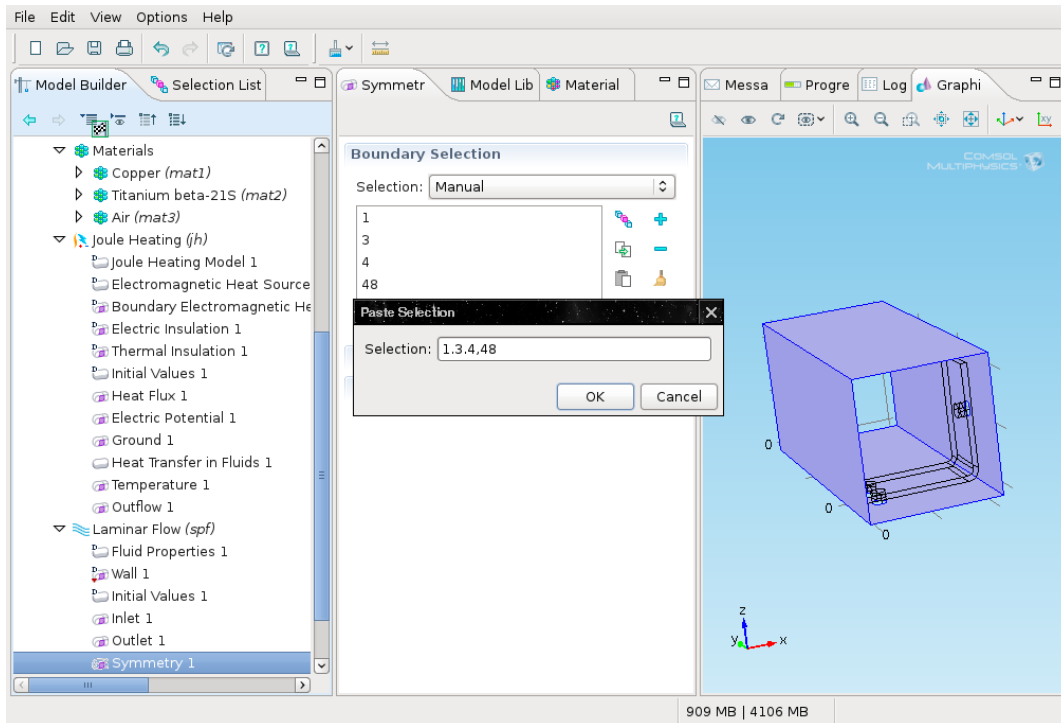


Figure 23: Entering the selection.

---

**Remark:** The indexes can be entered via the *Paste Selection* dialogue if the indexes are known (Fig. 23).

---

## Reducing the Mesh Resolution

In order to get a fast solution, the mesh resolution has to be lower. The present mesh parameters result in a considerable long time to calculate the solution but the mesh can always be refined (Fig. 24).

- 1 *Model Builder* → *Mesh 1* → *Size* → *Predefined = Normal*.
- 2 Click the *Build All* button.

## Simulation Sequences: Fluid and Joule Heating

It can assume that the flow speed is high enough to neglect the influence of the temperature rise to the flow field.

It concludes that the flow field can be solved first followed by the temperature problem using the solution of the flow field as its input.

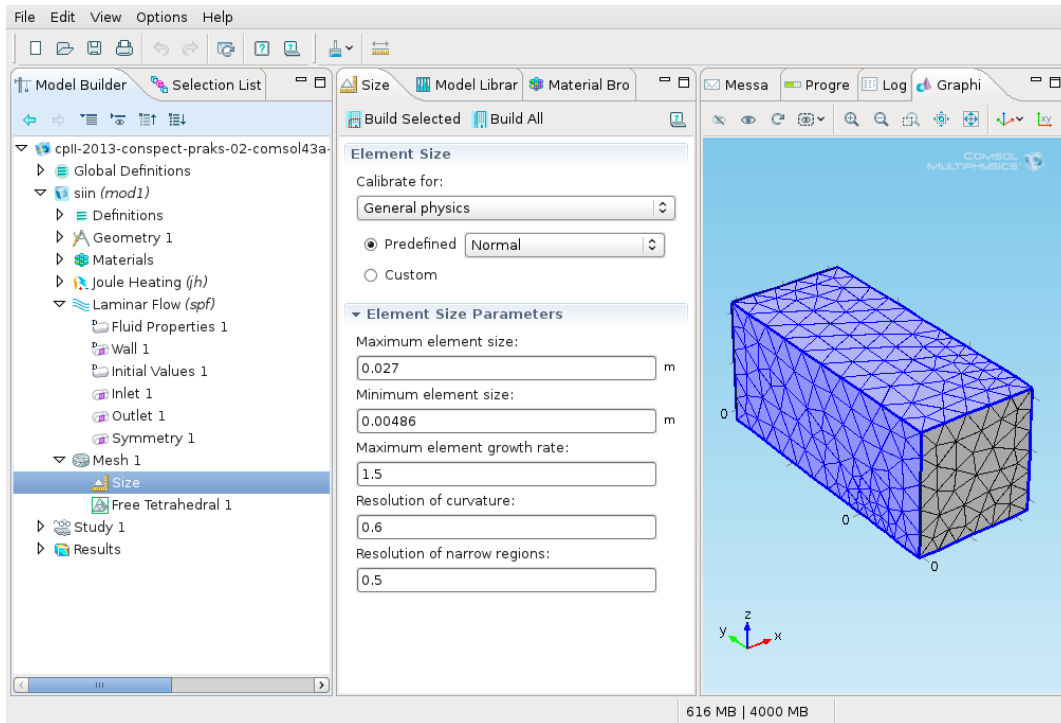


Figure 24: Mesh.

If the flow field is solved before the temperature, it becomes a weakly coupled multiphysical problem.

- 1 Add the second stationary study-step: **Model Builder** → **Study 1: right button** → **Study Steps** → **Stationary**.
- 2 Each study step should have correct physics interfaces active. Therefore remove the Joule Heating from the first study-step: **Study 1** → **Step 1 : Stationary** (Fig. 25).
- 3 Mark the **Joule Heating (jh)** as inactive.
- 4 Mark the **Laminar Flow (spf)** as inactive for the second study step of the **Study 1** (Fig. 26).
- 5 Run the simulation.
- 6 Turn the plot transparent after the run to visualise the temperature distribution inside the box.

The temperature plot (Fig. 27) shows the temperature in the busbar and in the space surrounding it. Since the mesh is quite coarse, the temperature field is not smooth. The reasonable way to get a smoother solution is to increase the mesh resolution.

- 7 Save the model into the **busbar-box\_1.mph** file for the future use if desired.

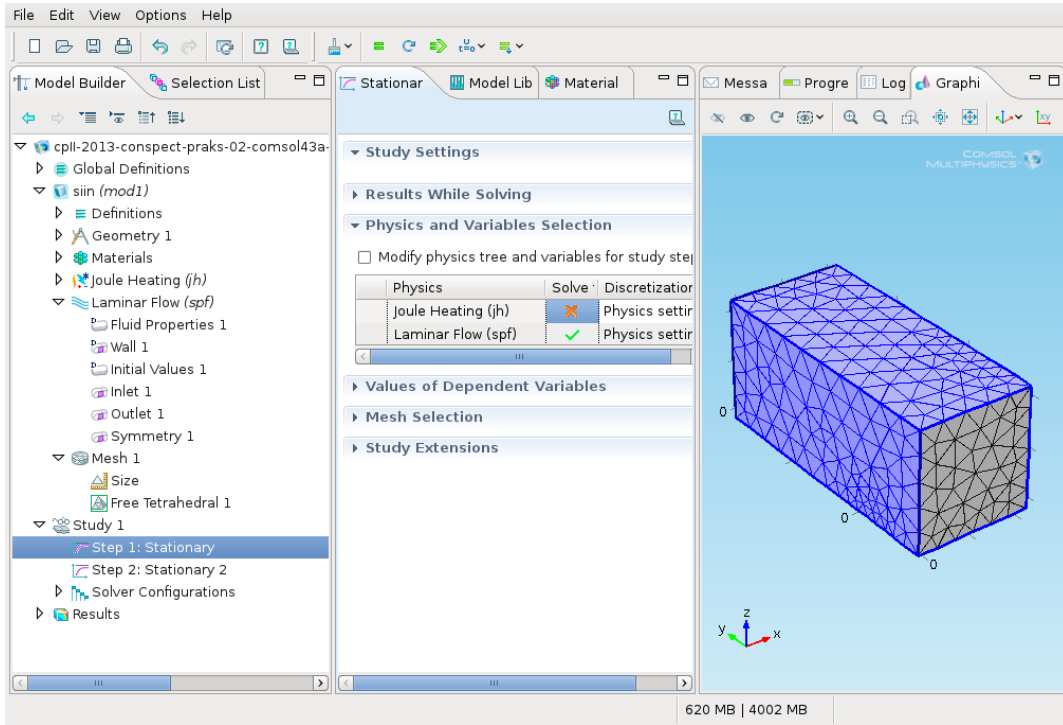


Figure 25: The parameters of the first study-step.

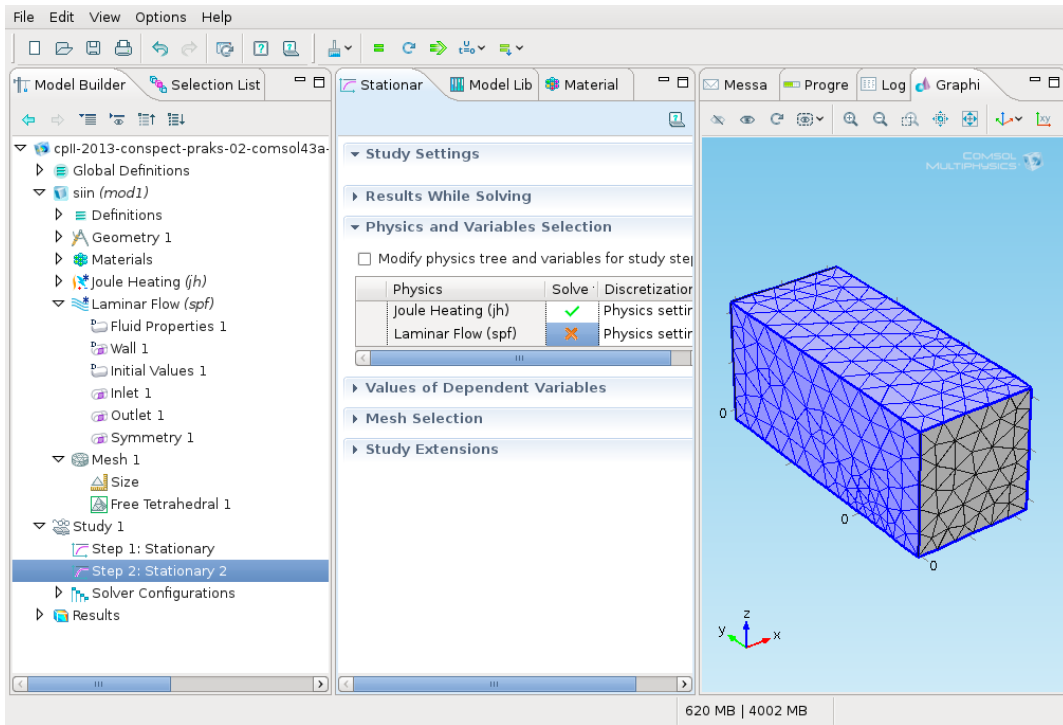


Figure 26: The parameters of the second study-step.

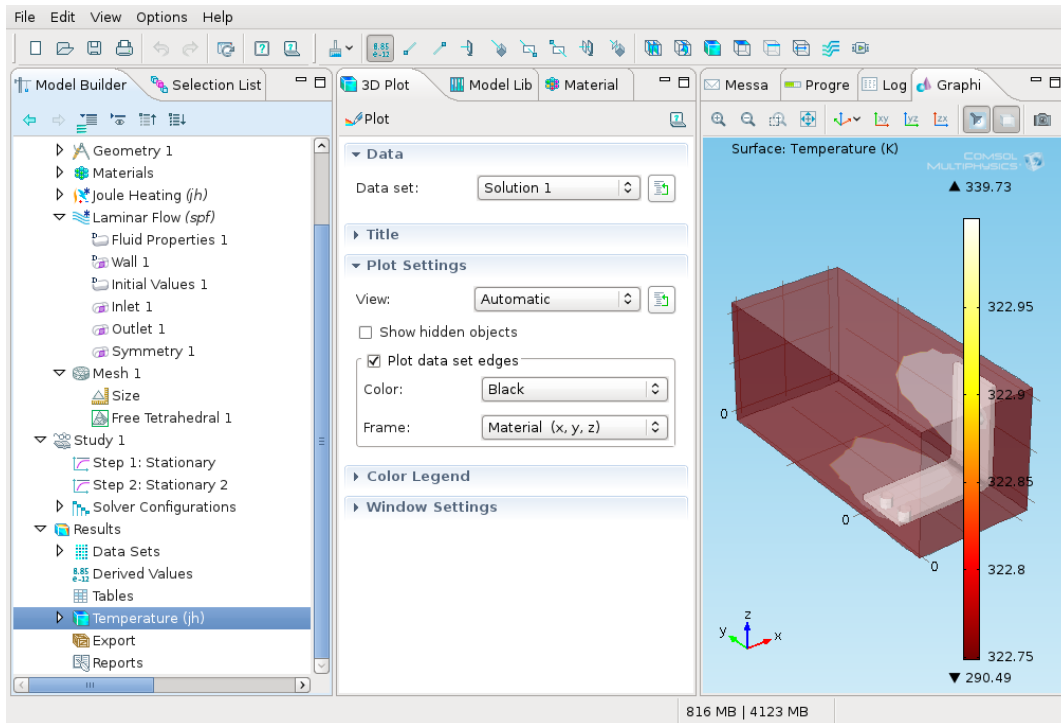


Figure 27: Temperature.

### Task 5:

- 1 Plot the temperature of the air box and the busbar.
- 2 Plot the flow speed as a cross-section in the  $xy$ -,  $zx$ -, and  $yz$ -planes in the same plot.

## Parametric Sweep

There is often a need and interest to create several versions of the model to investigate the effects of the different conditions. In the case of the busbar, the goal might be to decrease the working temperature. This can be achieved by reducing the current density. Since the current density depends on the geometry of the busbar, then changing the width  $w_{bb}$  should change the current density also and therefore influence the working temperature. This study can be done by the parametric sweep of the  $w_{bb}$ .

### Adding the Parametric Sweep

- 1 Open the file *busbar-solved.mph*. **Model Builder** → **Study 1: right button** → **Parametric Sweep** (Fig. 28).



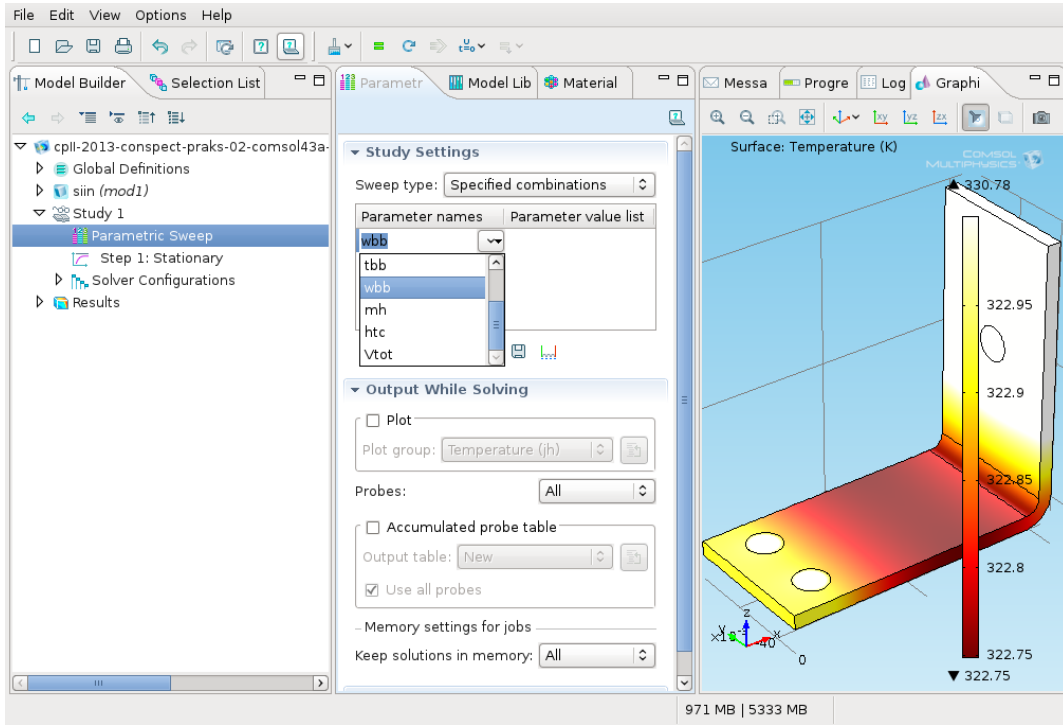


Figure 28: Adding the parametric sweep.

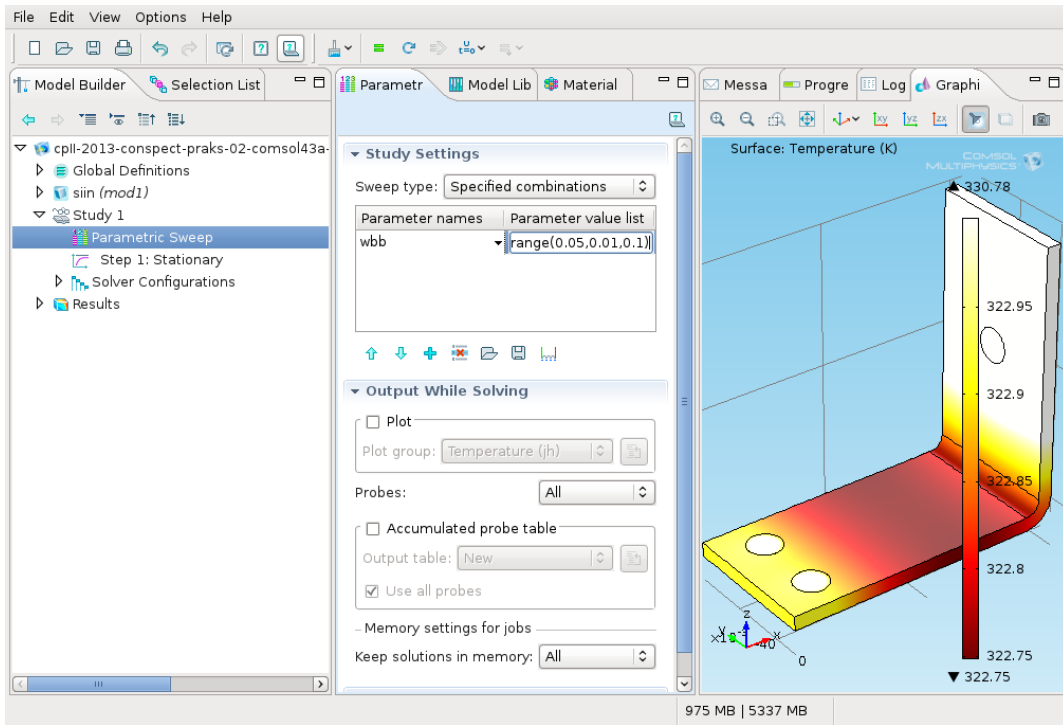


Figure 29: Entering the range as a function.

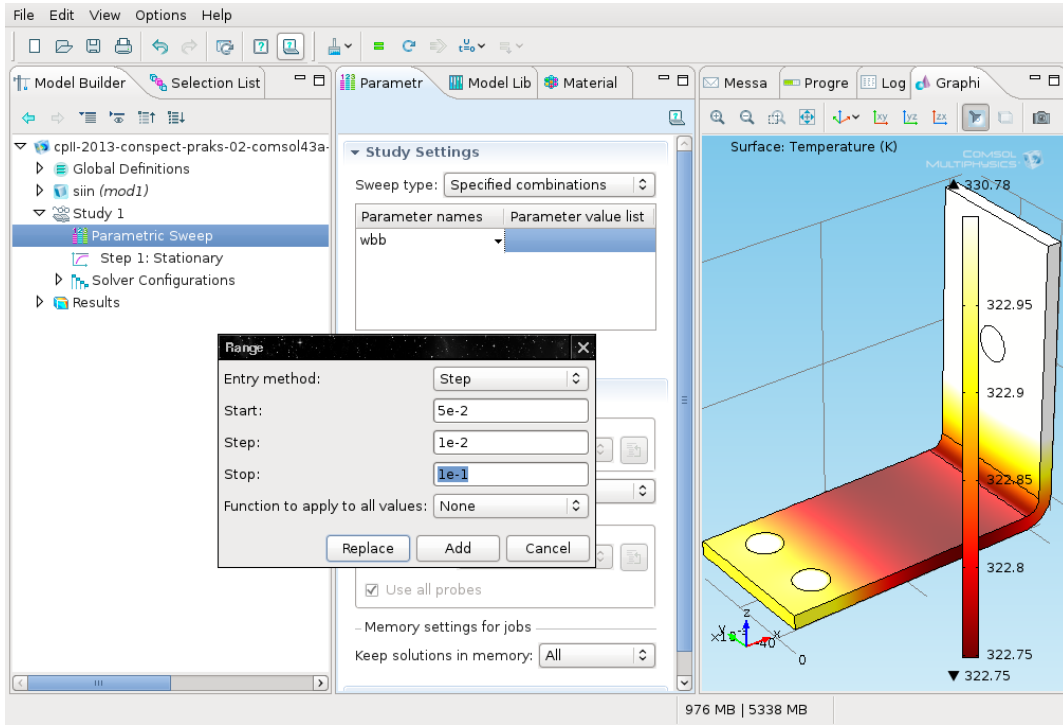


Figure 30: Range dialogue.

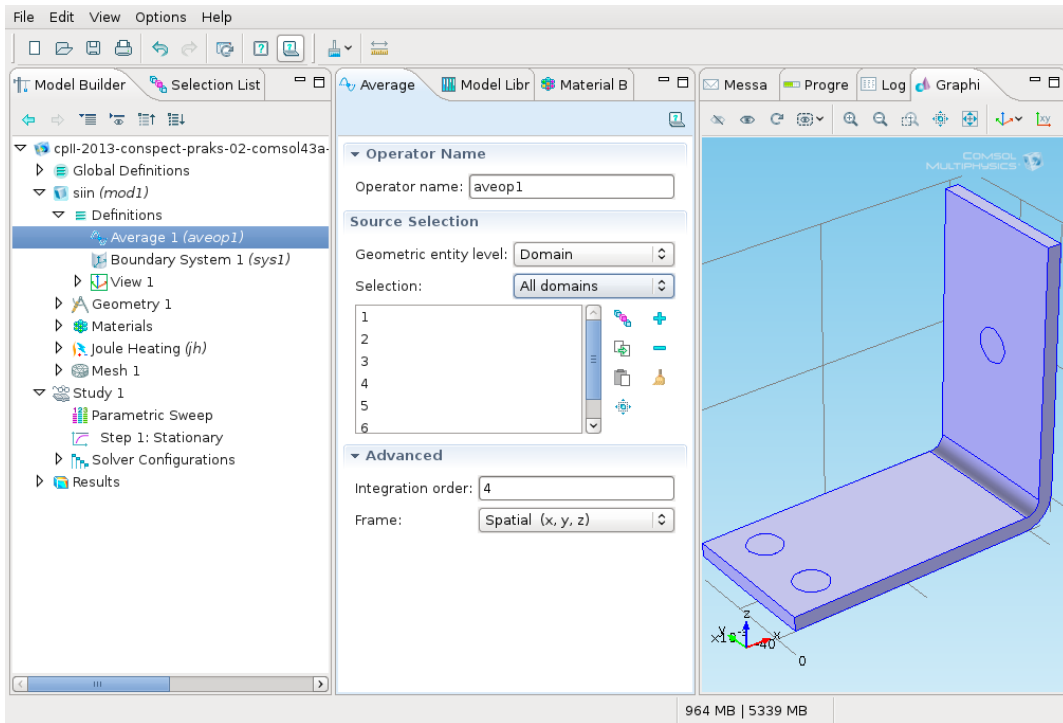


Figure 31: Creating the operator.

- 2 Click on the “+” button under the table in the parameters window and select *wbb* from the list appeared in the table under the “Name” column.
- 3 Enter the width range of 5 to 10 cm with the step of 1 cm for *wbb*. There are two ways to do it:
  - enter *range(0.05,0.01,0.1)* into the *Parameter value list* column (Fig. 29);
  - click on the *Range* button under the table and enter the values in the dialogue window (Fig. 30):
    - *Start = 5e-2*;
    - *Step = 1e-2*;
    - *Stop = 1e-1*;
 and click the *Replace* button.
- 4 Next, the *Average Model Coupling* is created for later use to calculate the average temperature: *busbar* → *Definitions: right button* → *Model Couplings* → *Average*. Select all domains (Fig. 31).
 

The operator *aveop1* makes possible to calculate the average value of the arbitrary quantity in the selected domains like the average value of the temperature, electric potential, current density etc.
- 5 Run the simulation.
- 6 Save the model into the *busbar\_III.mph* file.

## The Results of the Parametric Sweep

The *Temperature (jh) 1* branch is added into the *Results* section. The plot shows the temperature of the busbar for the last value in the parameter range, i.e. *wbb = 0.1 m (10 cm)* (Fig. 32). Since the color of the plot is quite uniform the limit of the maximum temperature has to be changed.

- 1 *Temperature (jh) 1* → *Surface* → *Range* → *Manual color range (check box)* → *Maximum = 309.35* (Fig. 33).
- 2 Compare the temperature plot of the wide busbar to the temperature plot of the narrow one (*wbb = 0.05 m*) (Fig. 34): *Model Builder* → *Results: right button* → *3D Plot Group*.
- 3 Select the *Solution 2* from the *Data set* list. This data array holds the results of the parametric sweep.
- 4 Select *0.05* from the *Parameter value (wbb)* list.
- 5 *Model Builder* → *Results* → *3D Plot Group: right button* → *Surface*. Click the *Plot* button.
- 6 To adjust the temperature limits closer to the wide busbar: *3D Plot Group 3* → *Surface 1* → *Range* → *Manual color range (x)* → *Maximum = 323* (Fig. 35).

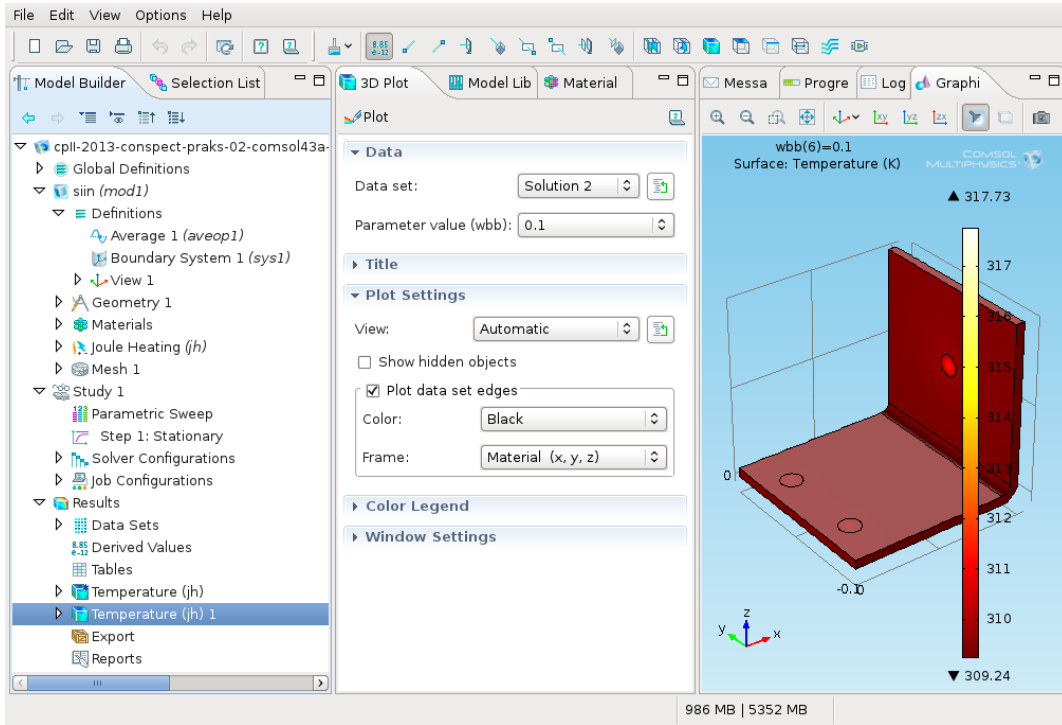


Figure 32: Temperature in the 10 cm wide busbar.

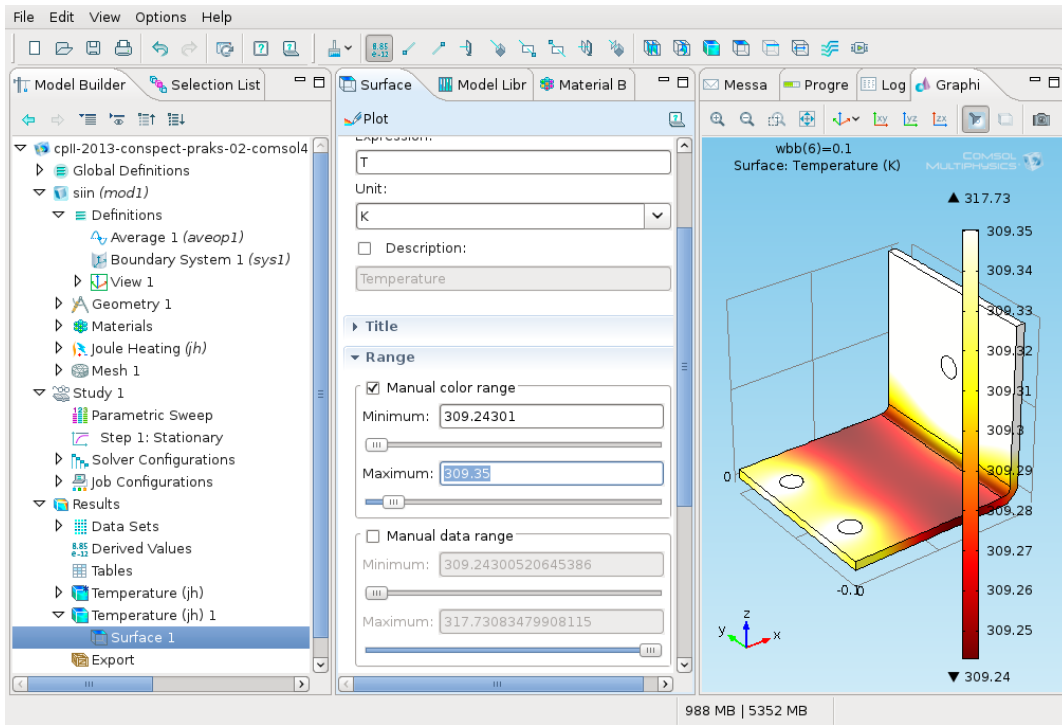


Figure 33: Changing the temperature plot for the 10 cm wide busbar.

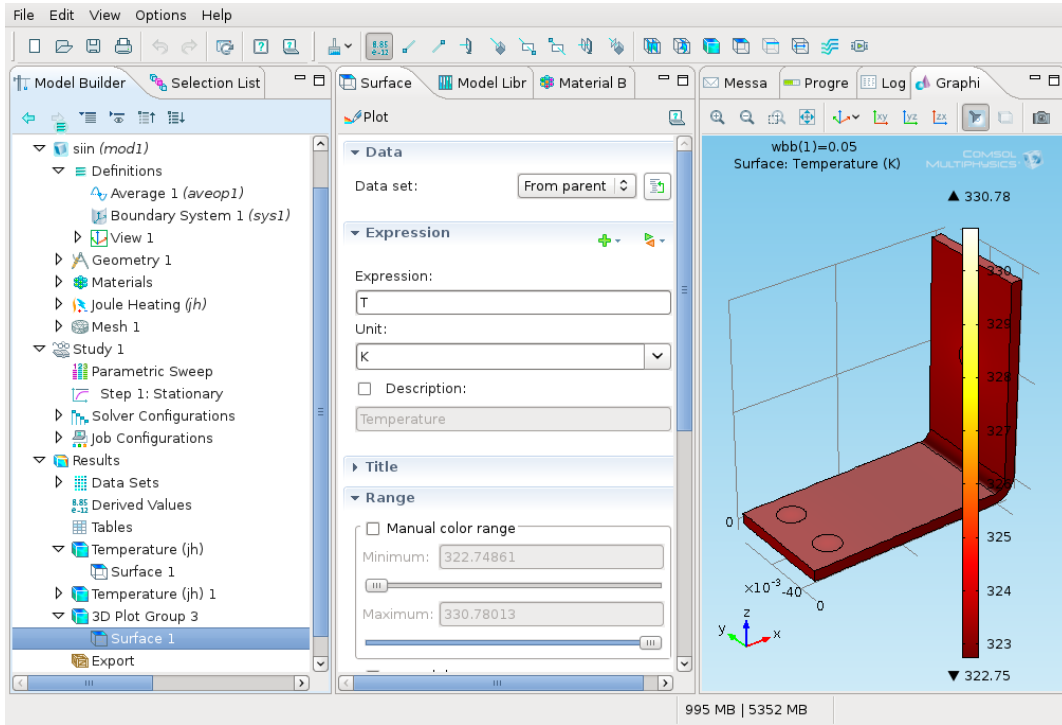


Figure 34: Temperature in the 5 cm wide busbar.

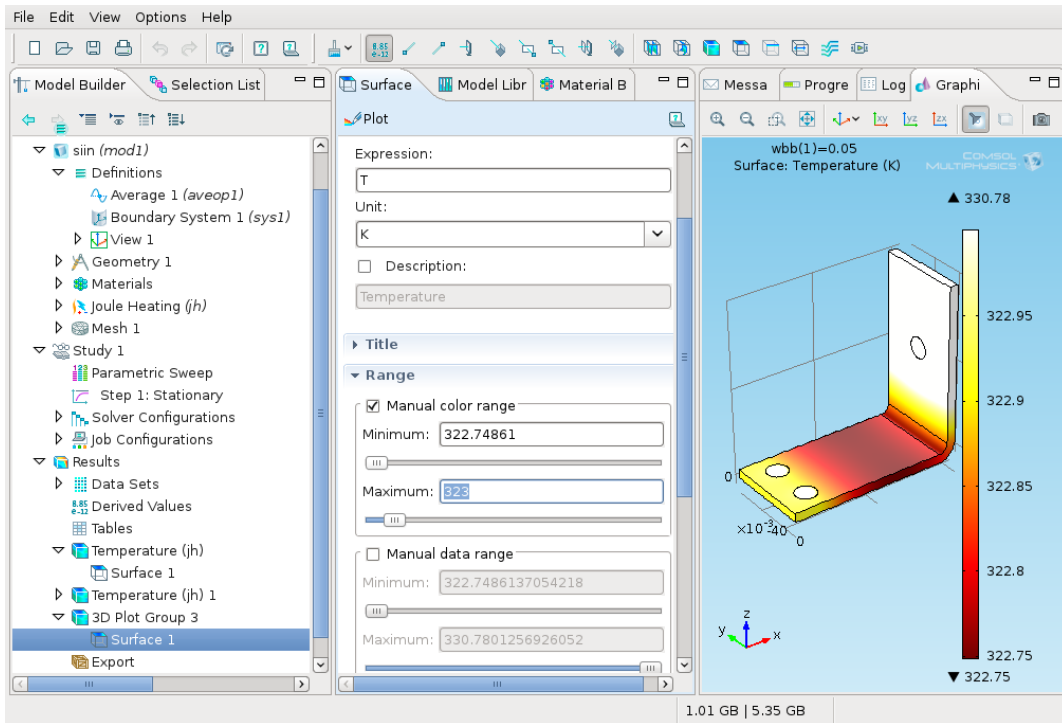


Figure 35: Changing the temperature plot for the 5 cm wide busbar.

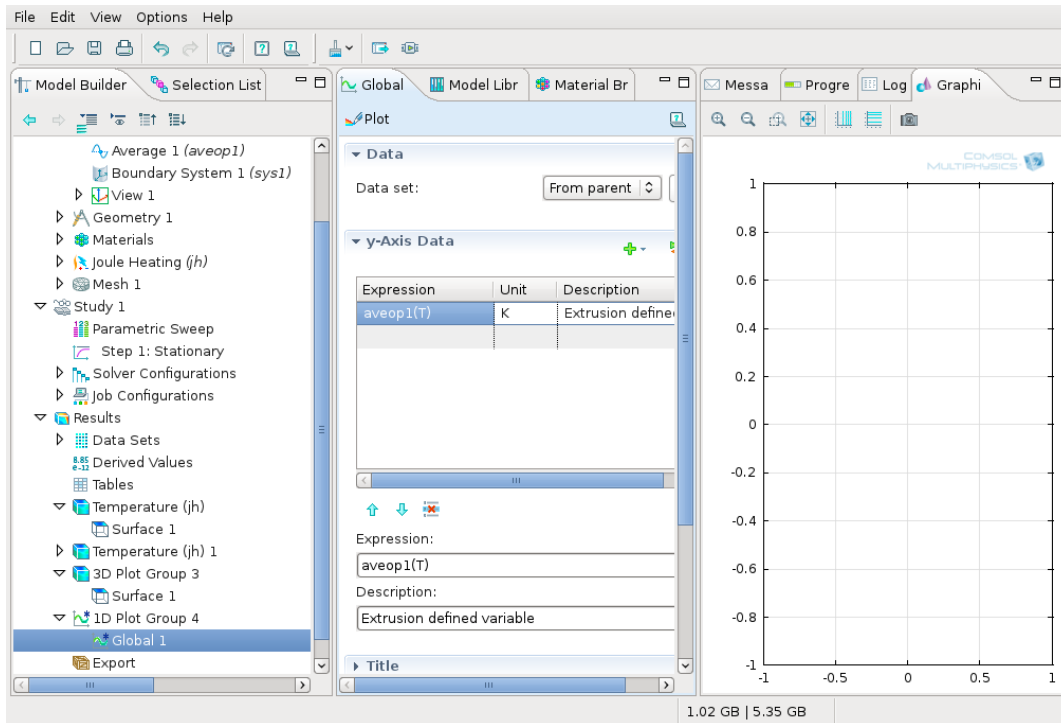


Figure 36: Using the average operator.

Comparing the temperature plots of the wide and narrow busbar it is seen, that the maximum temperature decreases from 331 K down to 318 K if the width of the busbar increases from 5 cm upto 10 cm.

---

**Task 6:** Plot the adjusted temperature for the 7 cm wide busbar.

---

## Adding the Plots

It is possible to plot the average temperature for every width.

- 1 **Results:** right button → **1D Plot Group**.
- 2 Activate **Solution 2** in the parameters window of the **1D Plot Group**.
- 3 **Model Builder** → **1D Plot Group:** right button → **Global**.
- 4 Add **aveop1(T)** into the first row of the **Expression** column in the **y-Axis Data** table (Fig. 36). The syntax remains similar while calculating the average values of the other quantities.
- 5 **Legends** → **Expression (x)** adds a legend in the upper right corner of the plot.
- 6 Click the **Plot** button.

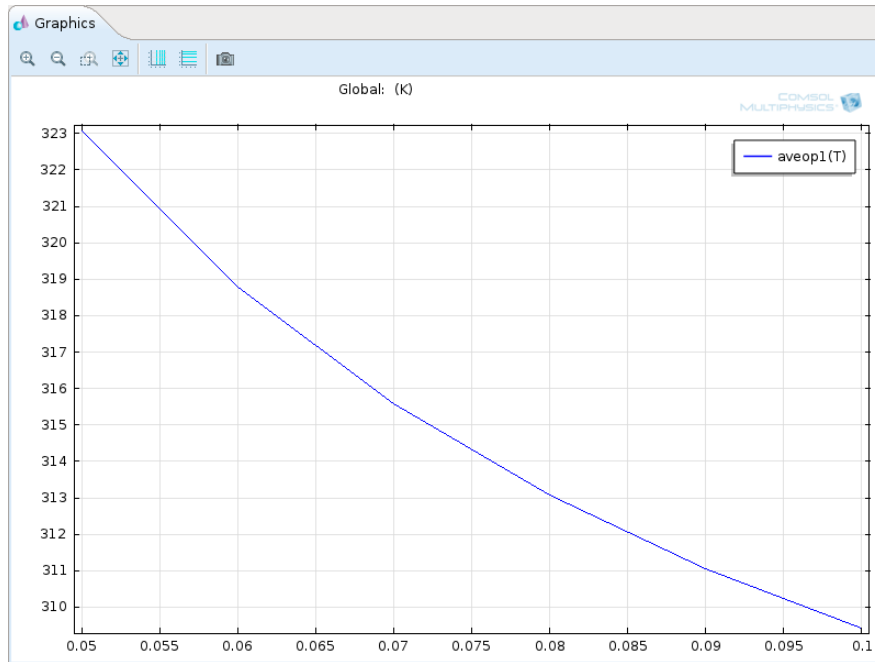


Figure 37: Average temperature as a function of the busbar width.

**7** Save the model into the busbar\_III.mph file.

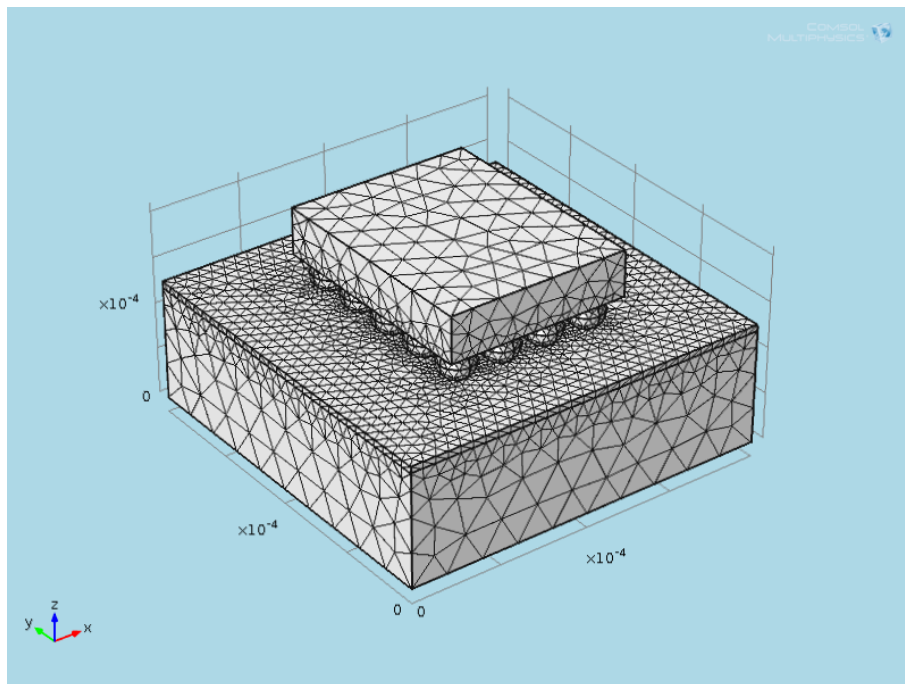
The plot (Fig. 37) shows that the average temperature decreases if the width of the busbar increases. Therefore, wider busbar helps to bring down the temperature. This probably increases the total mass (and the cost) of the busbar as a consequence. This may give a rise to some sort of the optimization problem.

---

**Task 7:**

- 1 Plot the average temperature as a function of the busbar width.
  - 2 Plot the average density of the busbar as a function of the busbar width.
  - 3 Plot the busbar mass as a function of the busbar width.
-

Computational Physics II  
Practical Work  
Comsol Multiphysics  
Mesh



Heiki Kasemägi

October 15, 2013



# Contents

- Mesh** **2**
- Introduction . . . . . 2
- Passing the Practical Work . . . . . 2
- References . . . . . 2
- Model Definition . . . . . 3
- Modelling . . . . . 3
- Geometry . . . . . 3
- Mesh 1 . . . . . 11
- Mesh 2 . . . . . 12
- Vörk 3 . . . . . 13
- Mesh 4 . . . . . 14
- Mesh 5 . . . . . 15
- Mesh 6 . . . . . 15
- Mesh 7 . . . . . 17
- Vörk 8 . . . . . 17
- Simple Square . . . . . 18

# Mesh

## Introduction

*COMSOL Multiphysics* contains an interactive meshing environment capable of meshing different parts of the geometry with individual approach with a little effort. Every meshing conception is added into the common meshing sequence and the final mesh is a result of the all operations in the meshing sequence.

The present example demonstrates how to create a mesh with different types of the elements. It gives the overview of adding, moving, disabling and deleting meshing steps and controlling the mesh via the size of the mesh elements.

## Passing the Practical Work

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading “**Task #**”. The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may rise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size, title(s), legend(s) etc. The report and accompanying files are submitted via moodle.

## References

“Using Meshing Sequences” in “COMSOL Multiphysics Model Library” Ver. 4.3.a.

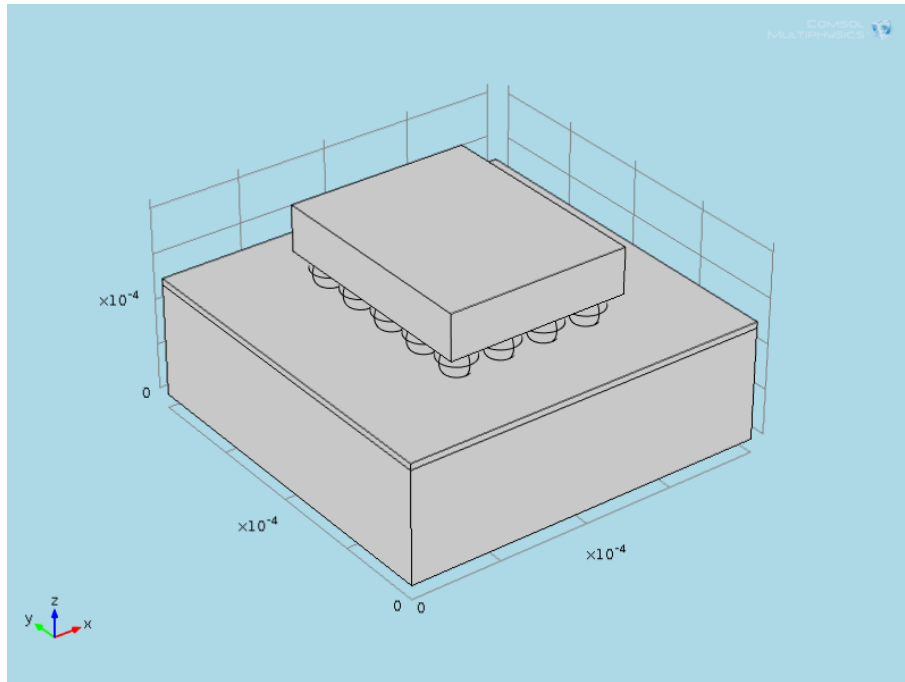


Figure 1: A piece of the circuit board with a chip on it.

## Model Definition

The geometry in the Fig. 1 represents a piece of the circuit board with a chip on it. The connectors between the chip and the board are the cylindrical solders. The measures of the details are in the Figures 2, 3 and 4.

Electronics components can have high temperatures for A relatively long time, therefore the permanent deformations and disconnections can occur due to the cracks in the solders.

## Modelling

### Geometry

- 1 To create the geometry start *COMSOL Multiphysics* or pick from the menu: **File** → **New**.
- 2 **Select Space Dimension: 3D** → **Finish**. Since there is no physics interface at the moment, continue with the model.
- 3 Add the bottom block (Fig. 5): **Model Builder** → **Geometry 1: right button** → **Block** → **Block 1**:
  - **Width** =  $bw$ ;
  - **Depth** =  $bw$ ;
  - **Height** =  $bh$ .

Click the **Build Selected** button.

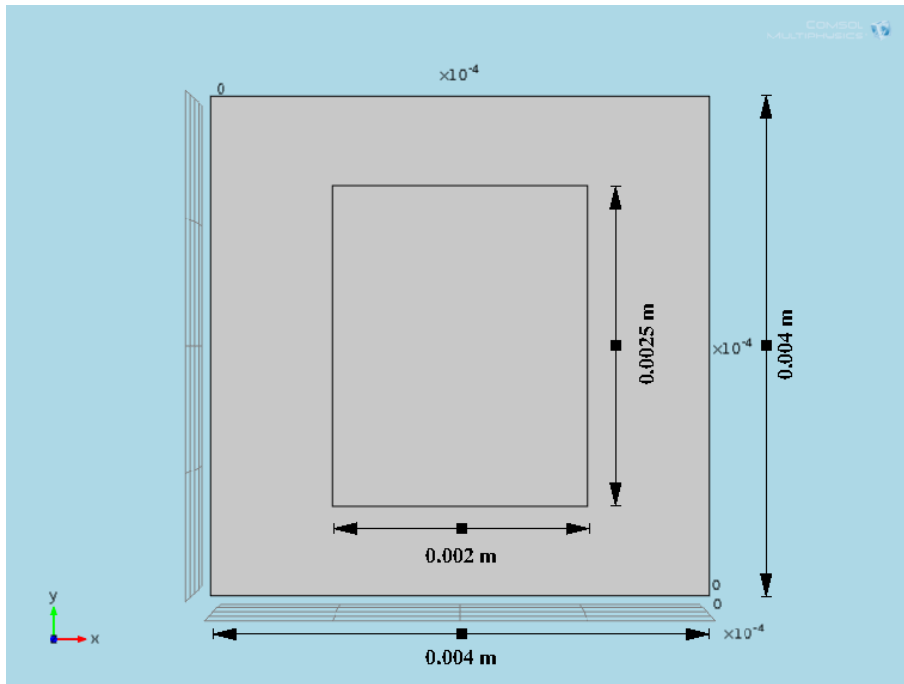


Figure 2: XY view.

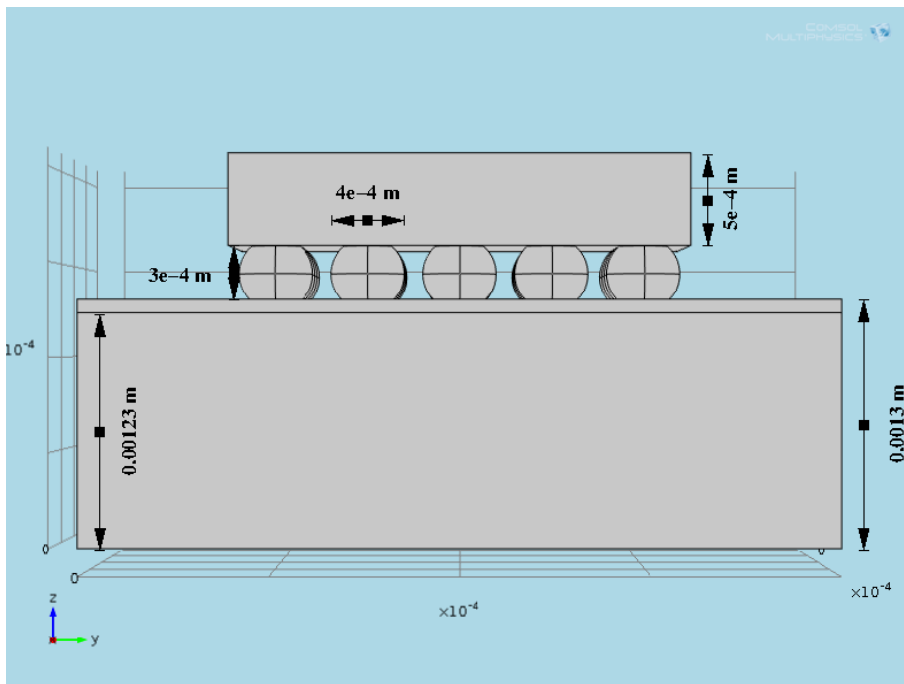


Figure 3: YZ view.

Parameters

Name	Expression	Value	Description
bw	0.004[m]	0.0040000 m	width of the bottom block
bh	0.00123[m]	0.0012300 m	height of the bottom block
sbh	7e-5[m]	7.0000E-5 m	height of the narrow block
dub	3e-4[m]	3.0000E-4 m	distance between the narrow and upper block
ubw	0.002[m]	0.0020000 m	width of the upper block
ubd	0.0025[m]	0.0025000 m	depth of the upper block
ubh	5e-4[m]	5.0000E-4 m	height of the upper block
ubx	0.001[m]	0.0010000 m	x-coordinate of the upper block
uby	7.5e-4[m]	7.5000E-4 m	y-coordinate of the upper block
rad	2e-4[m]	2.0000E-4 m	radius of the sphere
spx	ubx+2.5e-4[m]	0.0012500 m	x-coordinate of the sphere
spy	2.5e-4[m]+uby	0.0010000 m	y-coordinate of the sphere
spz	bh+sbh+dub/2	0.0014500 m	z-coordinate of the sphere
ubz	bh+sbh+dub	0.0016000 m	z-coordinate of the upper block
dispx	5e-4[m]	5.0000E-4 m	distance between adjacent solders in x-direction

Name:

Expression:

Description:

Figure 4: Measures of the components.

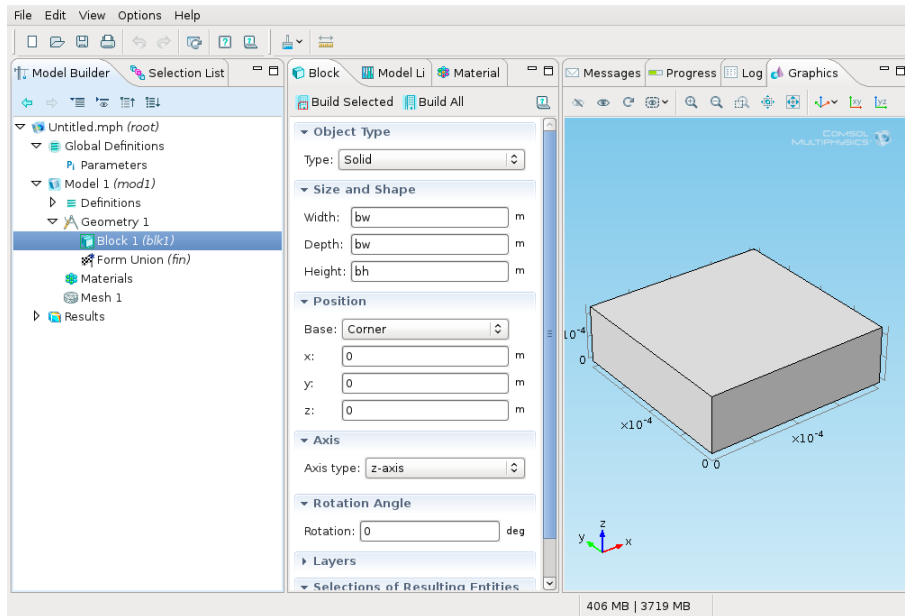


Figure 5: Bottom block.

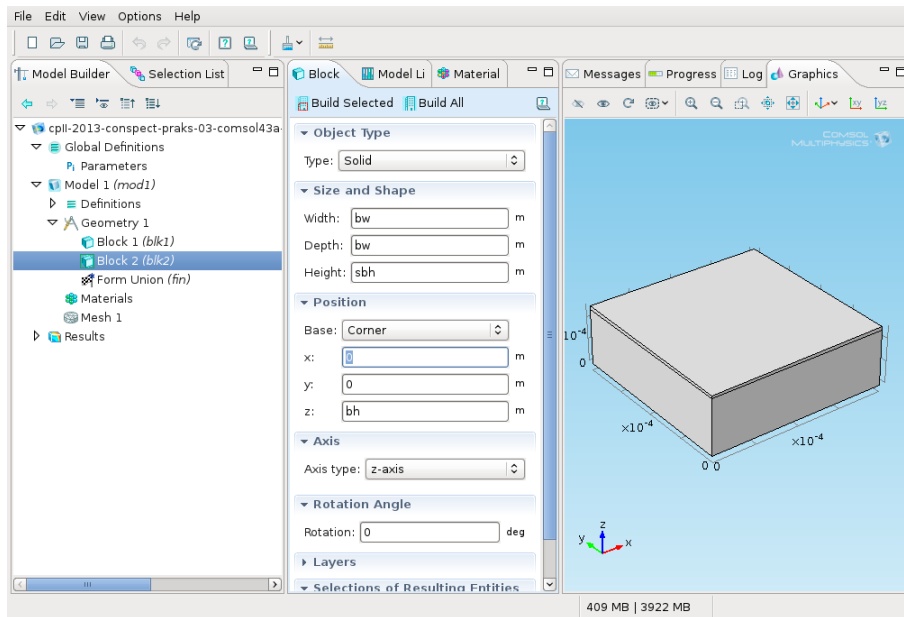


Figure 6: Narrow block.

4 Draw another, narrower block (Fig. 6): **Model Builder** → **Geometry 1: right button** → **Block** → **Block 2**:

- **Width** =  $bw$ ;
- **Depth** =  $bw$ ;
- **Height** =  $sbh$ ;
- **Base**:  $z = sbh$ .

Click the **Build Selected** button.

5 Draw the third block on top of the second one 0.0003 m from its surface (Fig. 7): **Model Builder** → **Geometry 1: right button** → **Block** → **Block 3**:

- **Width** =  $ubw$ ;
- **Depth** =  $ubd$ ;
- **Height** =  $ubh$ ;
- **Base**:  $x = ubx$ ;
- **Base**:  $y = uby$ ;
- **Base**:  $z = bh + sbh + dub$ ;

Click the **Build Selected** button.

6 The solder is a sphere with removed segments on the top and bottom in its nature (Fig. 8). To draw the solder: **Model Builder** → **Geometry 1: right button** → **Sphere** → **Sphere 1**:

- **Radius** =  $rad$ ;
- **X** =  $spx$ ;

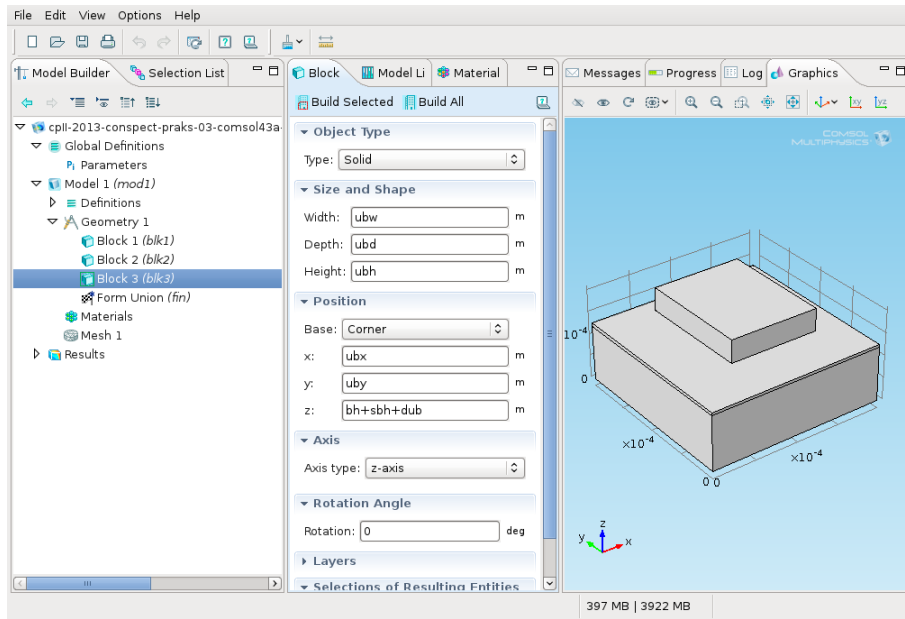


Figure 7: Chip block.

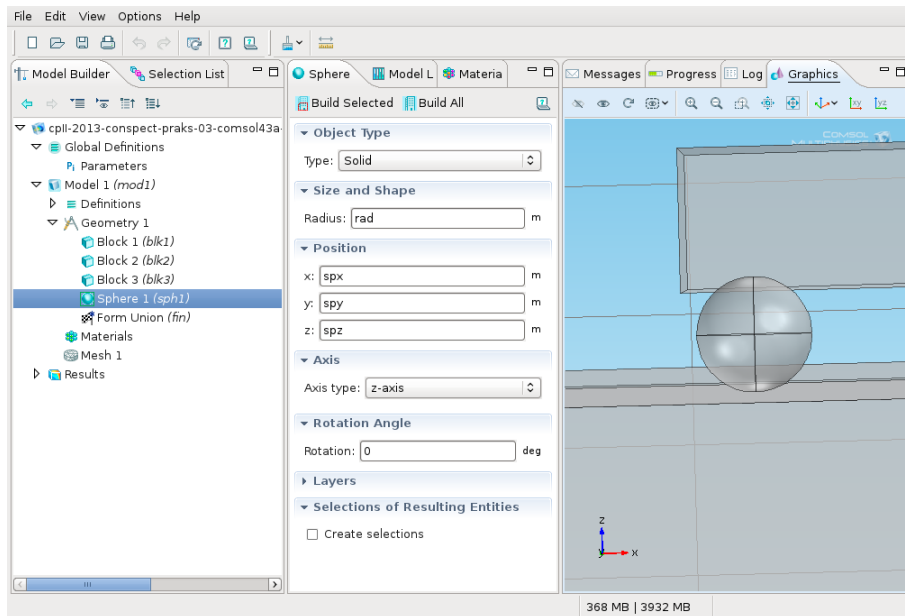


Figure 8: Sphere.

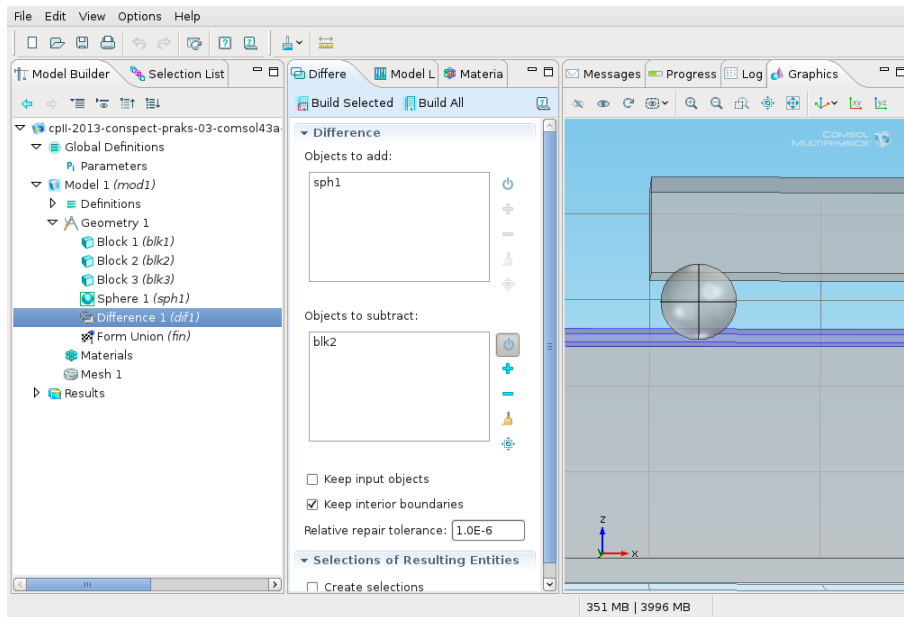


Figure 9: Removing the bottom segment of the sphere.

- $Y = spy;$
- $Z = spz.$

Click the **Build Selected** button.

Since the segments of the sphere intersect with the bottom and upper block, the segments need to be removed. To cut the bottom segment (Fig. 9):

- **Model Builder** → **Geometry 1: right button** → **Boolean Operations** → **Difference** → **Difference 1:**
  - **Objects to add: sph1;**
  - **Objects to subtract: blk2.**
- Click the **Build Selected** button.
- Removing the segment removes the narrow block also. To restore it (Fig. 10): **Model Builder** → **Geometry 1** → **Block 2: right button** → **Duplicate** → **Build Selected.**

7 Cut the top segment of the sphere in the same way and restore the upper block (Fig. 11).

8 The last step is to propagate the resulting solder (Fig. 12) propagate into 4-by-5 matrix with even distances between the elements (Fig. 13): **Model Builder** → **Geometry 1: right button** → **Transforms** → **Array** → **Array 1:**

- **Input objects: dif2;**
- **x-size = 4;**
- **y-size = 5;**
- **Displacement x = disp<sub>x</sub>;**



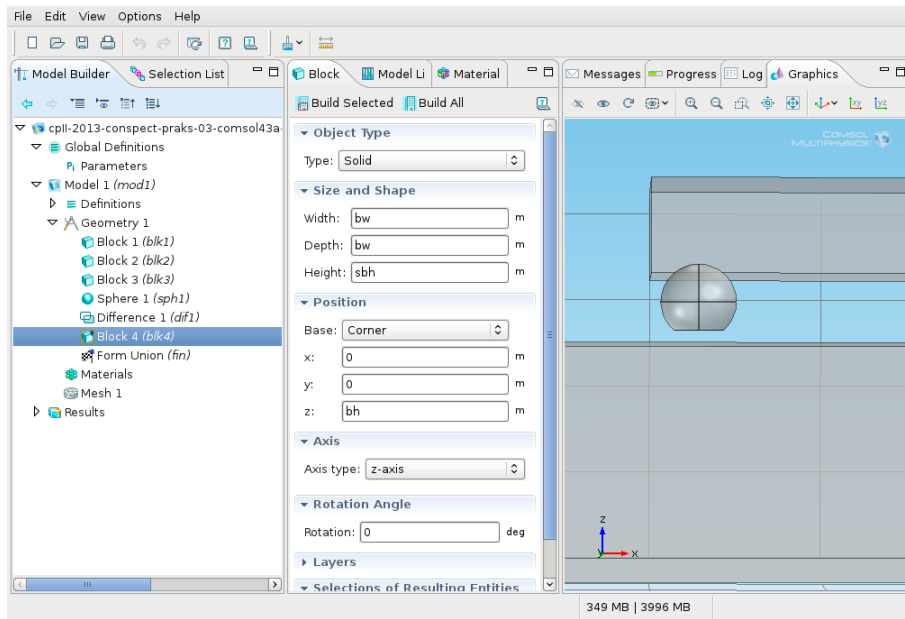


Figure 10: Restoring the narrow block.

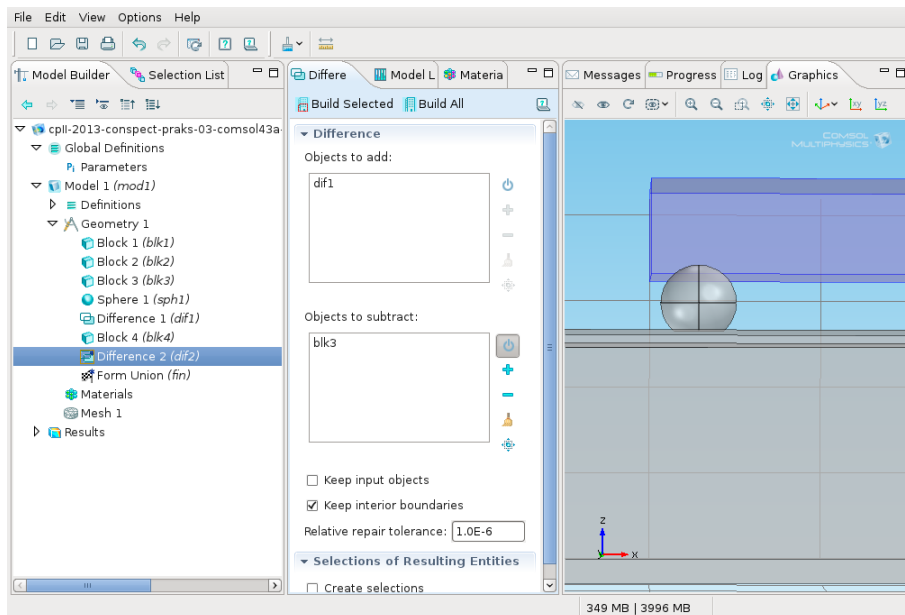


Figure 11: Removing the top segment.

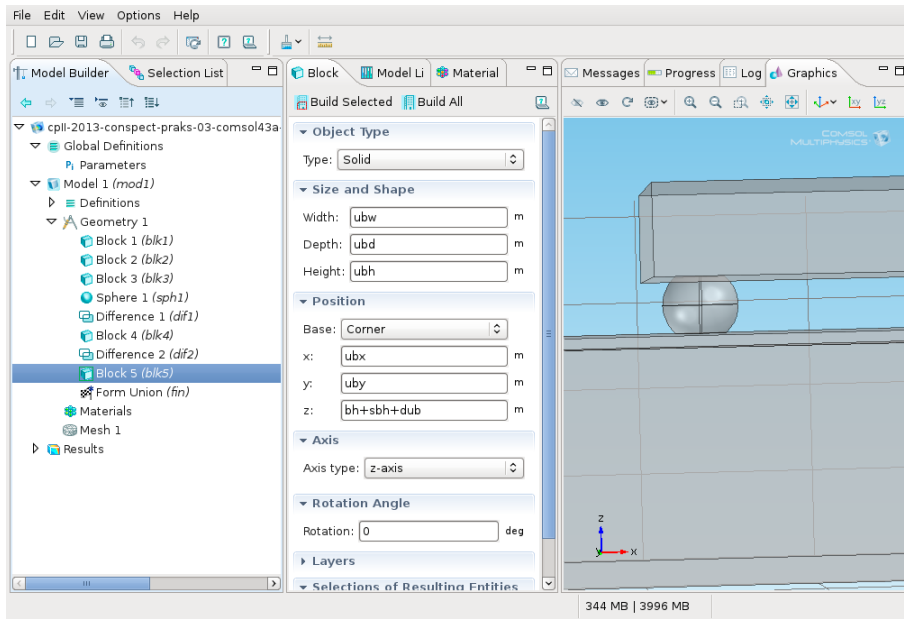


Figure 12: The solder.

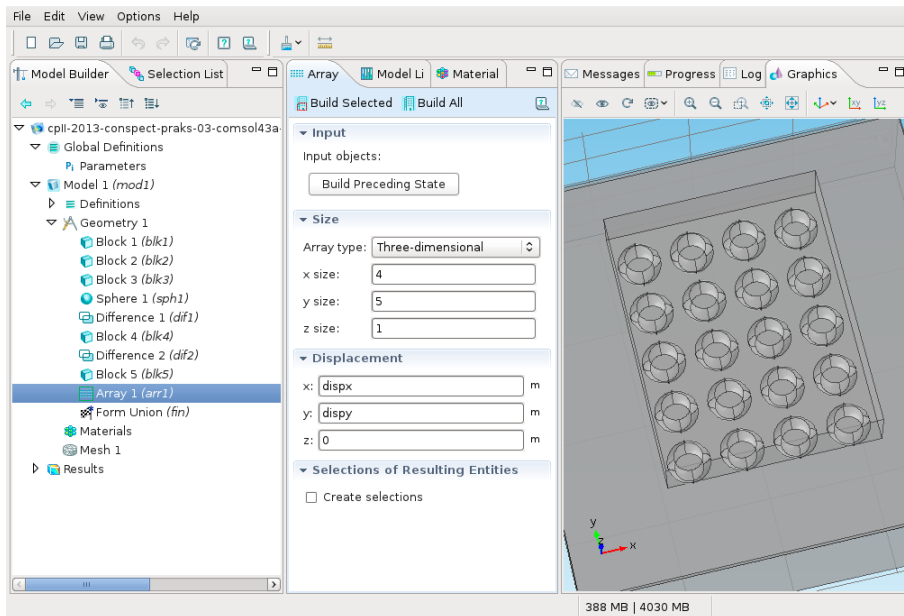


Figure 13: The array of solders.

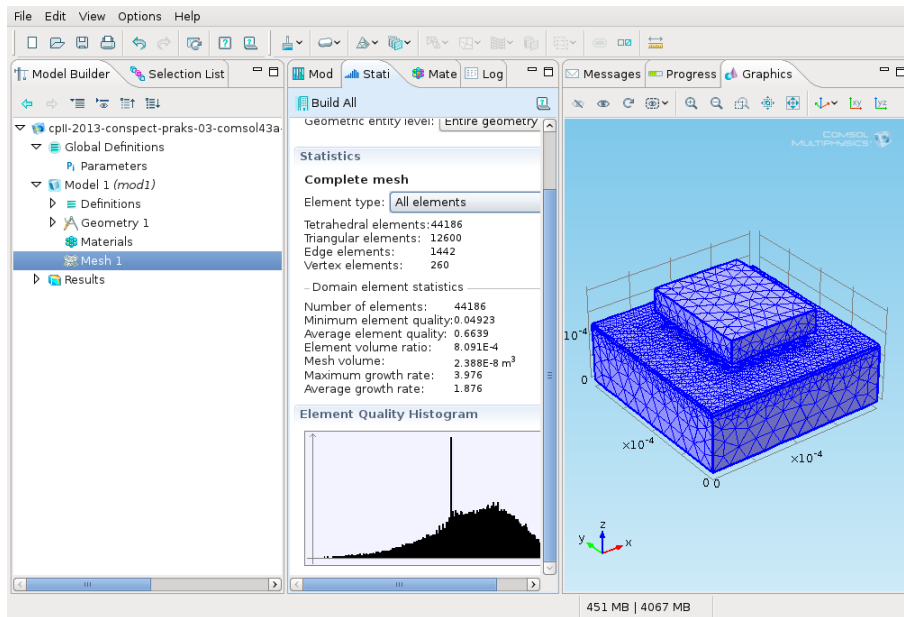


Figure 14: Default mesh.

- *Displacement  $y = dispy$ .*

Click the **Build Selected** button.

**Task 1:** Plot the geometry.

## Mesh 1

- 1 By clicking the **Model Builder** → **Model 1** → **Mesh 1** branch, the **Form Union** property is applied automatically, i.e. a union is created out of different geometrical objects in touch. The domains inside the union are separated by the inner boundaries. This ensures the possibility to create a continuous mesh between the domains, which are separate but in touch.

**Remark:** If **Sequence type** has the default value **Physics-controlled mesh** in the mesh parameters window, the **COMSOL Multiphysics** automatically generates the mesh corresponding to the parameters of the physics interface(s). In this case the mesh resolution has 9 predefined values ranging from **Extremely fine** down to **Extremely coarse**. The default value is **Normal**.

- 2 Click the **Build All** button.

The resulting mesh (Fig. 14) consists of about 44000 elements. Since it is quite a suitable for the model geometry, the number of elements may be reduced to reduce the amount of memory needed to solve the problem.

### 3 Set *User-controlled mesh* as a mesh type.

There are new default branches *Size* ja *Free Tetrahedral 1* in the model tree.

---

**Märkus:** *Size* has a global influence by affecting all subsequential meshing steps. This property is permanent and impossible to remove. The only way is to redefine it locally.

---

---

#### Task 2:

- 1 Present the mesh statistics: the number of the elements, the minimum and average quality of the elements in all three plate domains separately and solder domains all together. Are these parameters different in different solder domains? The line plots are preferred.
  - 2 Plot the quality of the volume elements.
  - 3 Plot the size of the volume elements.
  - 4 Is there any correlation between the quality and element size of the volume elements. If exists, describe it.
  - 5 Plot 10 % of the elements with the worst quality. How these are distributed in the domains?
  - 6 Plot 10 % of the elements with the best quality. How these are distributed in the domains?
- 

## Mesh 2

Lets keep the high resolution mesh in the solders, but decrease the mesh resolutions in other domains.

1 *Model Builder* → *Model 1* → *Mesh 1* → *Free Tetrahedral 1*: right button → *Size* → *Size 1*:

- *Element Size* → *Predefined* = *Coarser*;
- *Geometric entity level* = *Domain*;
- Add domains 1 . . 3.
- Click the *Build All* button.

The resulting mesh consists of about 28600 elements (Fig. 15).

To decrease the number of the elements even more, it is possible to create a swept mesh on the boundary and sweep it through the domain resulting in a structured mesh in the sweep direction.

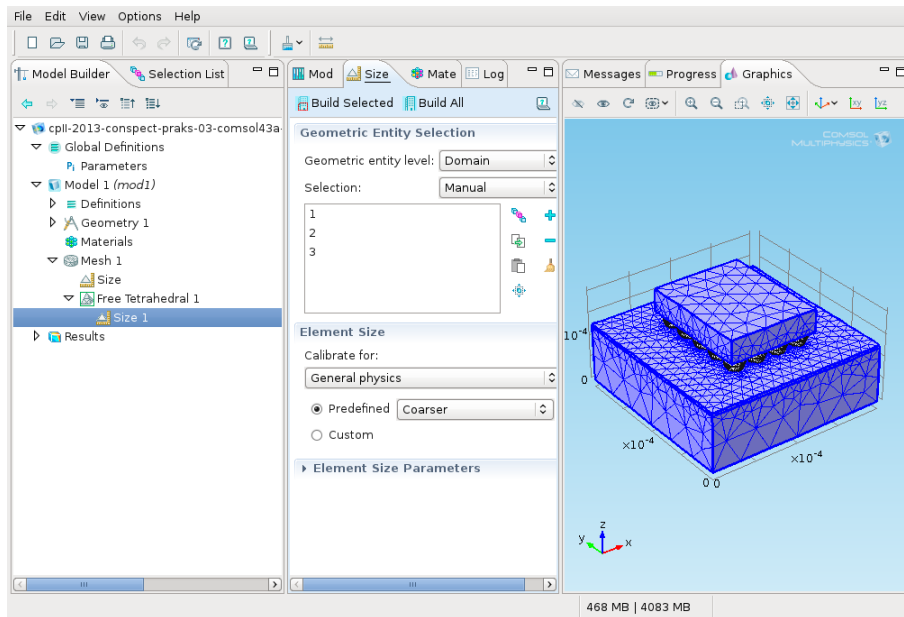


Figure 15: Reducing the mesh resolution locally.

---

### Task 3:

- 1 Plot the resulting mesh.
  - 2 What are the changes in elements minimum and average quality compared to the previous mesh?
- 

### Vörk 3

To change the mesh of the solders, one need to edit the tetrahedral mesh and deactivate corresponding *Size 1* branch since it is not needed any more.

- 1 ***Model Builder*** → ***Model 1*** → ***Mesh 1*** → ***Free Tetrahedral 1*** → ***Geometric entity level = Domain***.
  - 2 Select domains ***4 . . . 23*** by removing domains ***1 . . . 3*** from the selection list (Fig. 16).
  - 3 Click the ***Build All*** button.
  - 4 ***Model Builder*** → ***Model 1*** → ***Mesh 1*** → ***Free Tetrahedral 1*** → ***Size 1: right button*** → ***Disable***.
- 

### Task 4:

- 1 Plot the solder mesh as a mesh quality plot.
-

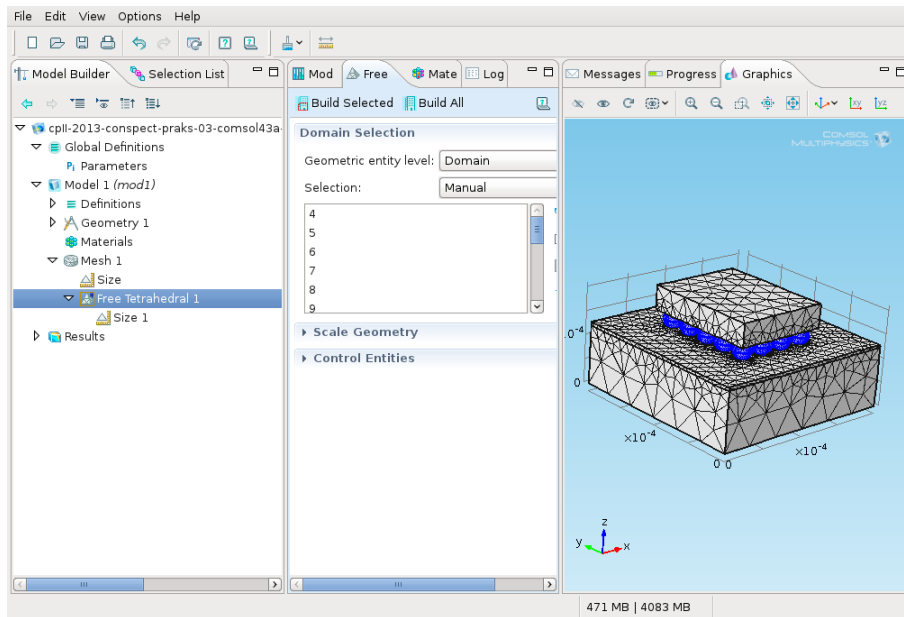


Figure 16: Selecting the solders.

## Mesh 4

The swept mesher first meshes a source face in the 3D domain and then sweeps the resulting face mesh through the domain to an opposite destination face. For straight and circular sweep paths several connected faces can be used as the source faces. All of the faces enclosing the domain can be graded as the source faces, a destination faces or linking faces. The latter link source and destination faces together.

In this case the sources of the swept mesh of the domains **2** and **3** (upper part of the circuit board and the chip) consist of several faces. By meshing faces **7** and **12** the complete source mesh for the following sweep mesh can be constructed.

**1 Mesh 1: right button → More Operations → Free Triangular.**

---

**Remark:** *Free Triangular 1* is inserted after the *Free Tetrahedral 1* (Fig. 17). *COMSOL Multiphysics* always inserts the new branches after the current branch. The current branch is marked by the box around the icon in front of the branch name. The inserted branch becomes instantly the new current branch.

It is easy to change the order of meshing by moving the branches in the meshing sequence. Care has to be taken if the next branch depends on the results of the previous one.

---

**2 Model Builder → Model 1 → Mesh 1 → Free Triangular: right button → Move Up.**

**3 Model 1 → Mesh 1 → Free Triangular 1: right button → Move Down.**

**4** Click the *Wireframe Rendering* button in the rendering window.

**5** Select boundaries **7** and **12**.

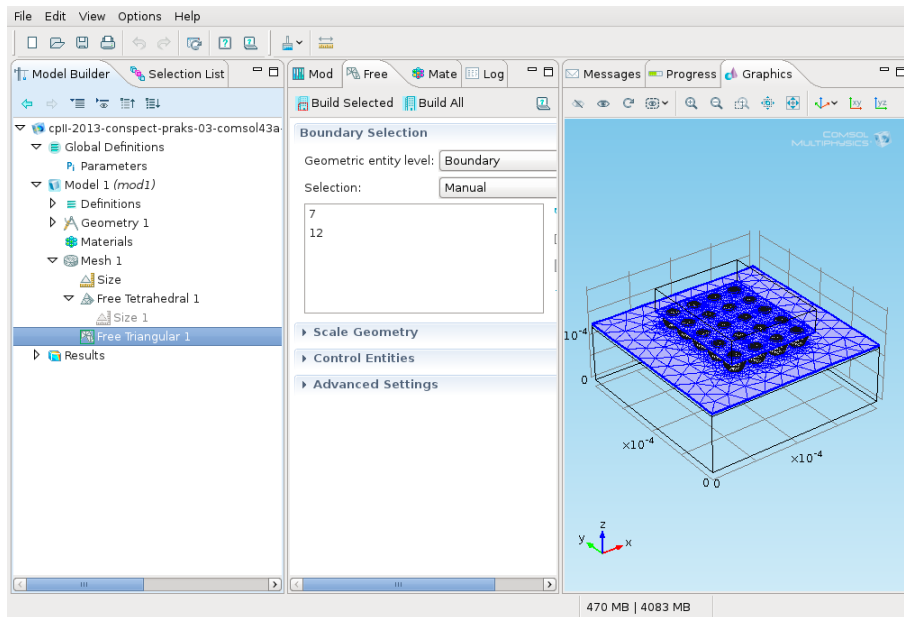


Figure 17: Adding triangular mesh branch.

6 Click the **Build All** button.

---

### Task 5:

1 Plot the mesh.

---

## Mesh 5

While creating the triangular mesh, the global element size parameter was considered. To coarser the mesh, the local size parameter has to be introduced into the free triangular mesh settings.

- 1 **Model 1** → **Free Triangular 1**: right button → **Size**.
- 2 **Free Triangular 1** → **Size 1** → **Element Size: Predefined = Coarser**.
- 3 Click the **Build All** button.

The resulting mesh (Fig. 18) looks almost like the previous one. The reason is that the local size parameter affects only the interior of the meshed faces. The mesher has to accept the existing mesh on the edges of the solder joints and the global size parameter on the outer edges which is set to normal mesh size.

## Mesh 6

To avoid the situation described in the previous section, it is a good practice first to set the global **Size** parameter value to the coarsest mesh what is planned to use for this geometry and then define a local size parameter for every meshing step requiring finer mesh.

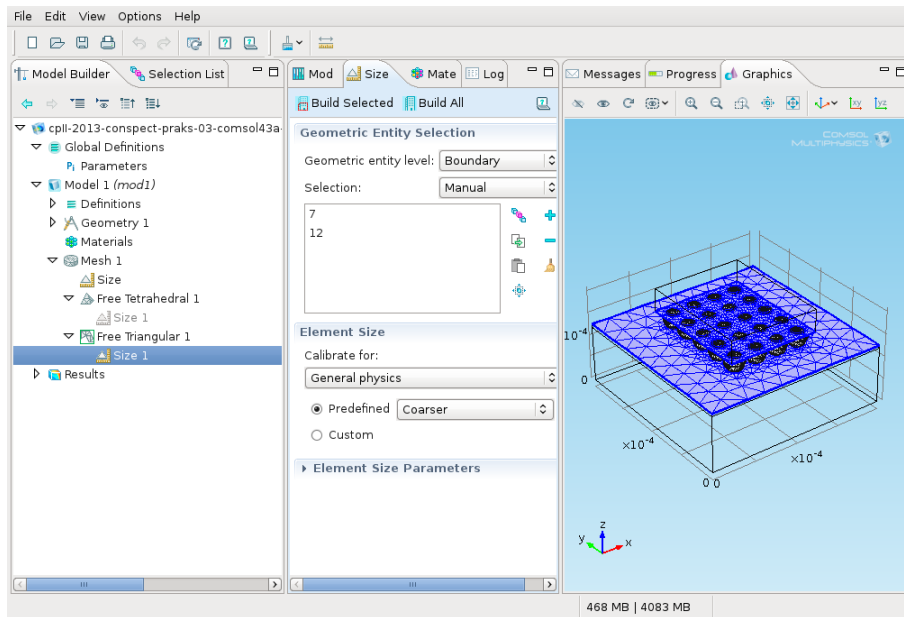


Figure 18: The resolution of the triangular mesh.

- 1 **Model Builder** → **Model 1** → **Size** → **Element Size: Predefined = Coarser**.
- 2 Since the global mesh size is now too large for the solder joints, the **Size 1** parameter has to be enabled in the **Free Tetrahedral 1** branch: **Model Builder** → **Model 1** → **Free Tetrahedral 1** → **Size 1: right button** → **Enable**.
- 3 **Free Tetrahedral 1** → **Size 1** → **Selection = All domains**.
- 4 Select domains **1-3** from the selection list ja click the **Remove from Selection** button.
- 5 **Free Tetrahedral 1** → **Size 1** → **Element Size** → **Predefined = Normal**.
- 6 Triangular mesh does not need the size parameter any more and it can be removed: **Model Builder** → **Model 1** → **Free Triangular 1** → **Size 1: right button** → **Delete**.
- 7 Click the **Build All** button.

The resulting triangular mesh has lower resolution this time (Fig. 19).

---

### Task 6:

- 1 Compare the statistics of the present and previous meshes separately for boundaries **7** (upper side of the thin plate facing the chip plate) and **12** (bottom side of the chip plate facing the thin plate).
-



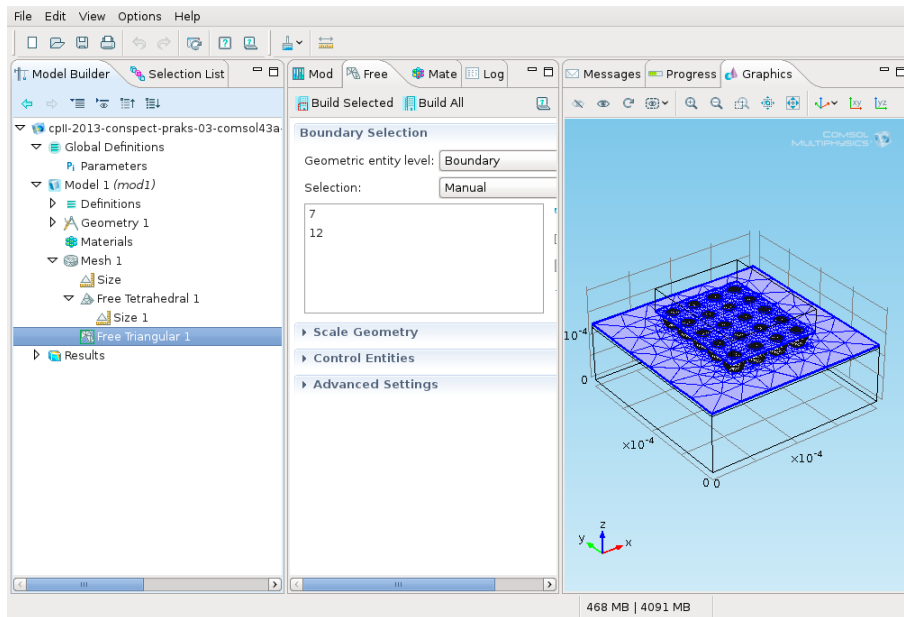


Figure 19: Coarser triangular mesh.

## Mesh 7

The next step is to sweep the source mesh through the rest of the domains.

- 1 **Model Builder** → **Mesh 1**: right button → **Swept**.
- 2 The default value of the **Geometric entity level** is **Remaining**. In the present case this corresponds comfortably to the domains to be meshed with the sweep mesh.
- 3 Click the **Build All** button.

The mesh (Fig. 20) now consists of about 16800 elements. The swept mesh profits from the predefined *Coarser* setting to determine the number of the elements along the sweep direction.

## Võrk 8

For more precise control of the element number one has to specify the distribution of the swept mesh elements.

- 1 **Model 1** → **Mesh 1** → **Swept 1**: right button → **Distribution**.
- 2 **Distribution 1** → **Number of elements** = 2.
- 3 Click the **Build All**-button.

The resulting mesh (Fig. 21) contains about 20450 elements, having the higher resolution in the domains what are more important for the analysis.

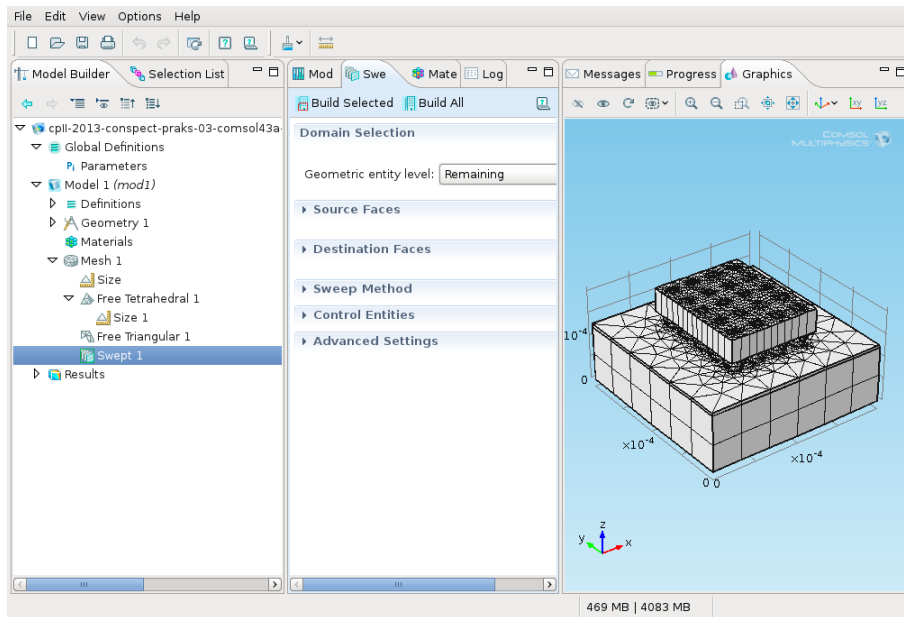


Figure 20: The swept mesh.

---

### Task 7:

- 1 Plot the mesh.
  - 2 What are the qualitative and quantitative differences compared to the previous mesh, if any?
- 

## Simple Square

The purpose of the following exercise is to get acquainted to the different aspects of the meshing on a simple square as an object. The side-length of the square is  $1\text{ m}$ .

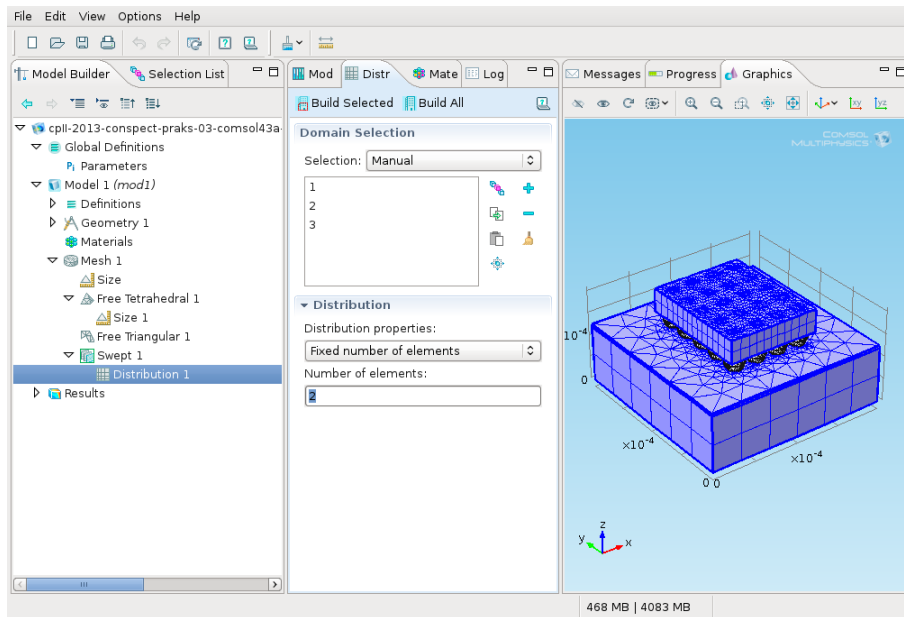


Figure 21: Modifying the swept mesh.

### Task 8:

- 1 Mesh the square with the user defined mesh in its default settings. Plot the number of the triangular elements, minimum and average quality as the functions of the type of the predefined mesh resolutions *Model Builder*  $\rightarrow$  *Mesh 1*  $\rightarrow$  *Size*  $\rightarrow$  *Element Size*  $\rightarrow$  *Predefined* as separate line plots.
- 2 Mesh the same geometry with quadrilateral mesh in its user defined default settings. Plot the number of the triangular elements, minimum and average quality as the functions of the type of the predefined mesh resolutions *Model Builder*  $\rightarrow$  *Mesh 1*  $\rightarrow$  *Size*  $\rightarrow$  *Element Size*  $\rightarrow$  *Predefined* in the plots of the task 8.1 correspondingly.
- 3 Compare the minimum and average element quality of the triangular mesh to the quadrilateral ones in the case of the predefined resolutions. What are the conclusions?
- 4 Mesh the same square with 4 equilateral elements according to Fig. 22. Plot the mesh as the element quality plot.
- 5 Mesh the same square with 16 equal elements in 4-by-4 arrangement according to the Fig. 23. Plot the mesh as the element quality plot.
- 6 Mesh the same square with 4 vertical elements with the same width. The length of the element is the length of the square. Plot the mesh as the element quality plot.
- 7 Mesh the same square with 25 elements. The arrangement of the elements is 5-by-5. The ratio of the side lengths of the first and the last element along the axis is 2 according to the Fig. 25. This applies to the both axes. Plot the mesh as the element quality plot.
- 8 Compare the element quality plots of tasks 8.4, 8.5, 8.6 and 8.7 to each other. What are the conclusions about the element size, geometry and quality?

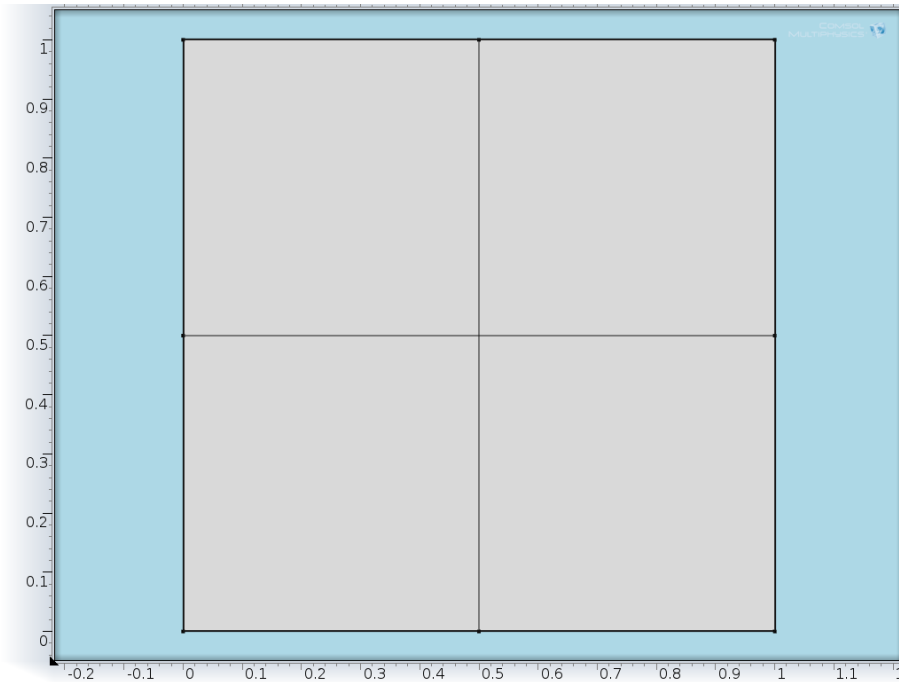


Figure 22: 2-by-2 mesh.

---

**Task 9:**

- 1 Draw a semi-arc with outer radius of 1 m and thickness of 0.2 m.
  - 2 Mesh the semi-arc's left half with the uniform quadrilateral elements in two rows having the same width. Each row should have 20 elements.
  - 3 Mesh the semi-arc's right half with two rows of the elements, 5 elements per row. The ratio of the side lengths of the first (right from the top of the semi-arc) and the last (right end of the semi-arc) elements should be 2 (Fig. 26).
  - 4 Plot the mesh as the elements quality plot and give the conclusions about the elements location, size and quality.
-

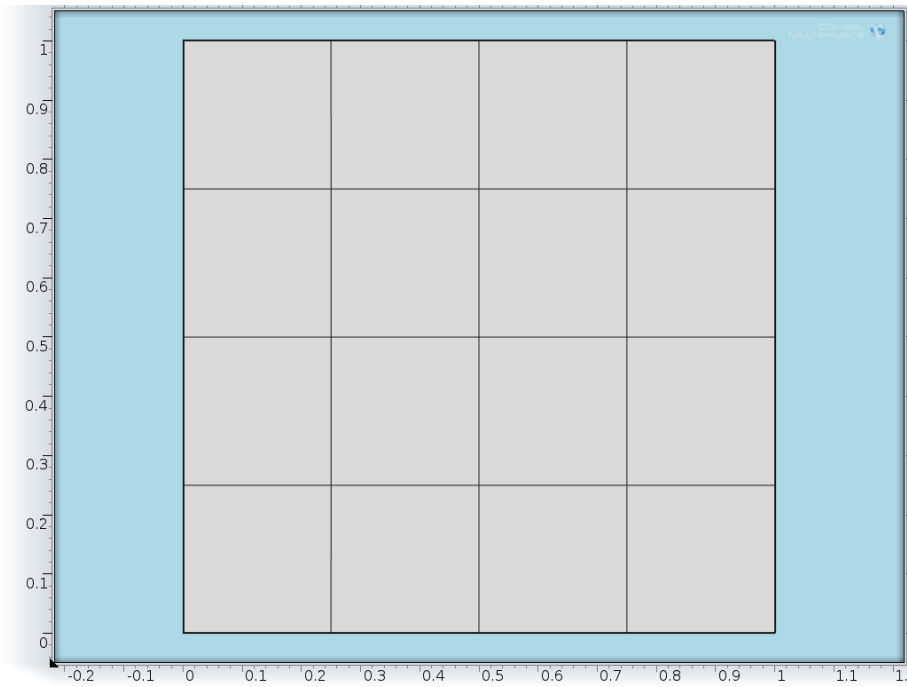


Figure 23: 4-by-4 mesh.

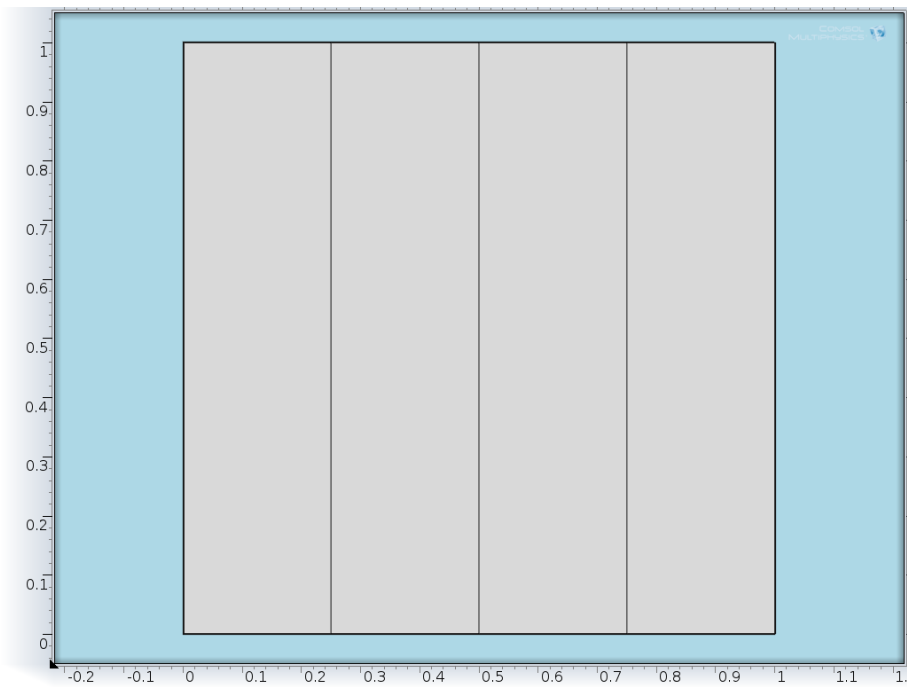


Figure 24: 4-by-1 mesh.

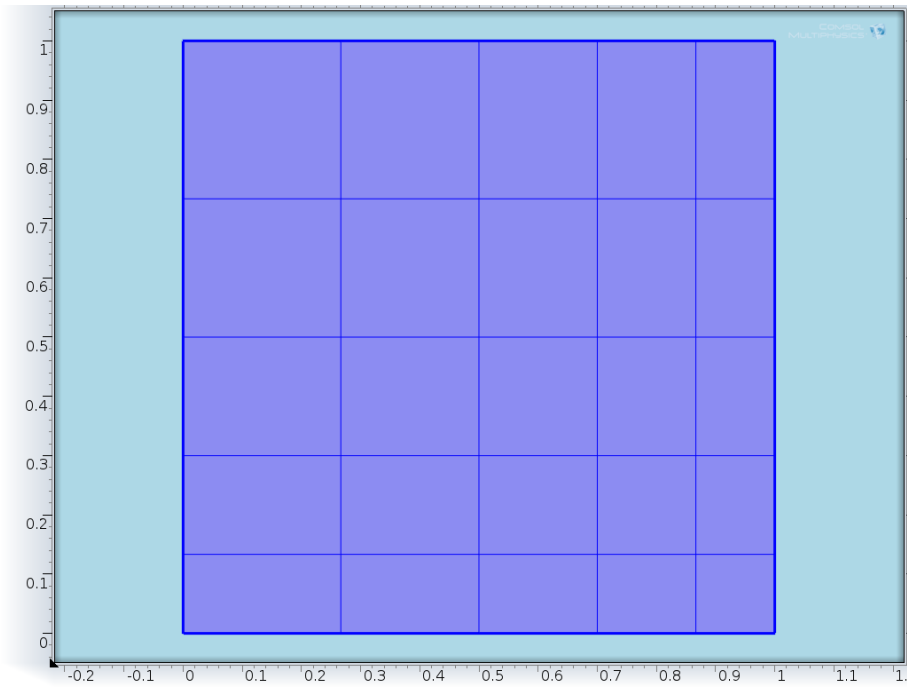


Figure 25: Convergent mesh.

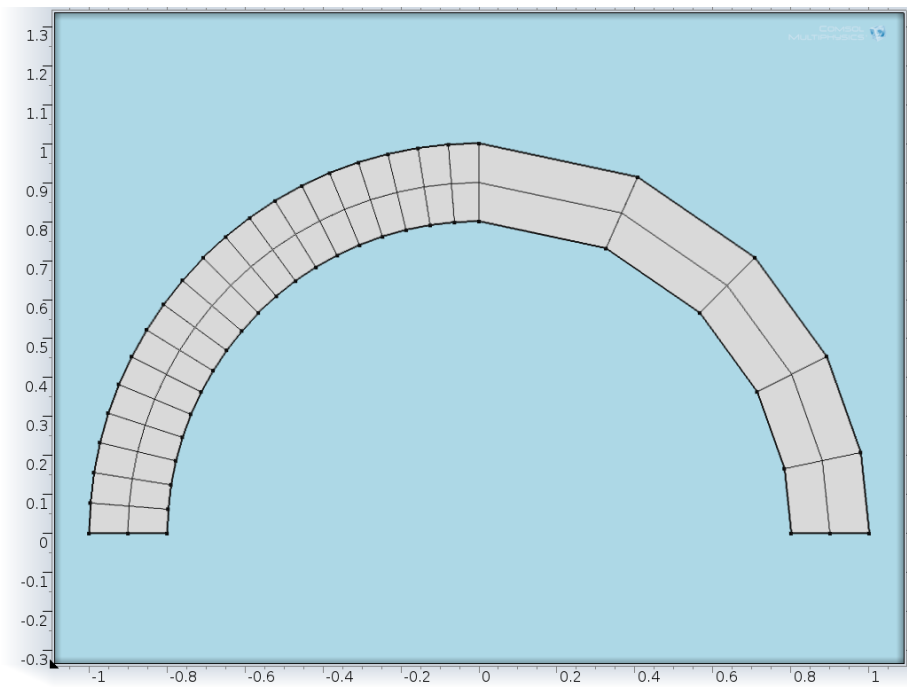
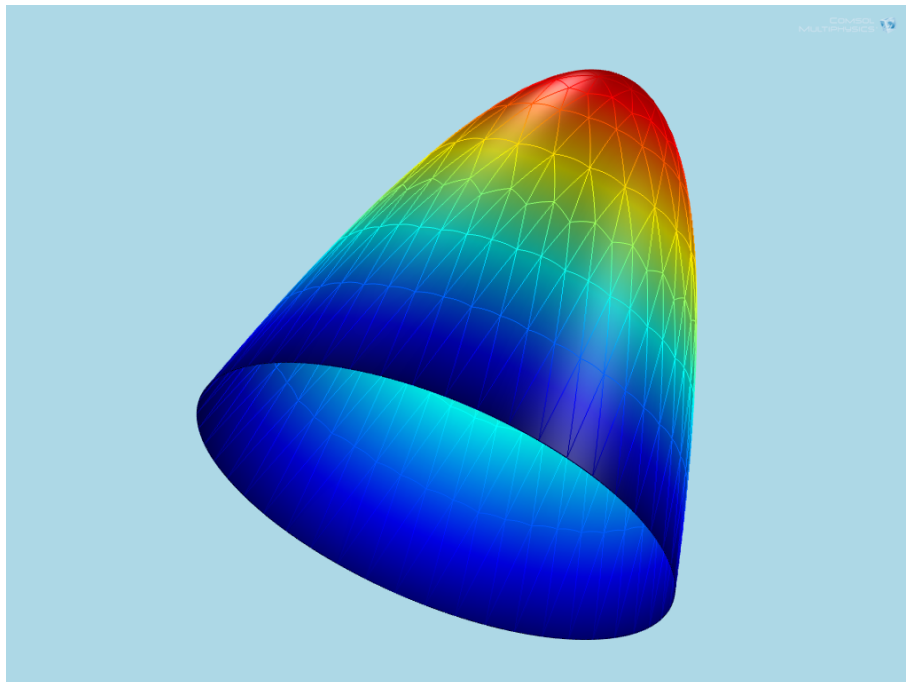


Figure 26: Semi-arc.

Computational Physics II  
Practical Work  
Comsol Multiphysics  
Partial Differential Equations



Heiki Kasemägi

October 14, 2013

# Contents

<b>Partial Differential Equations</b>	<b>2</b>
Introduction . . . . .	2
Passing the Practical Work . . . . .	2
References . . . . .	2
PDE Interface . . . . .	3
Coefficient Form . . . . .	3
General Form . . . . .	5
Poisson Equation . . . . .	6
Pellet . . . . .	8
Spherically Symmetrical Transport . . . . .	8
Model Definition . . . . .	8
Domain Equations . . . . .	8
Boundary and Initial Conditions . . . . .	10
Modelling . . . . .	10



# Partial Differential Equations

## Introduction

*COMSOL Multiphysics* has a set of modules to solve abstract Partial Differential Equations (PDEs) presented in “pen-and-paper” way. This combines the flexibility of the traditional “pen-and-paper” method and the engine and computing power of the modern Finite Element algorithms.

## Passing the Practical Work

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading “**Task #**”. The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may arise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size, title(s), legend(s) etc. The report and accompanying files are submitted via moodle.

## References

“Equation Based Modelling” in “COMSOL Multiphysics Modeling Guide” Version 3.5.

“Spherically Symmetric Transport” in “COMSOL Multiphysics Model Library” Version 3.5.

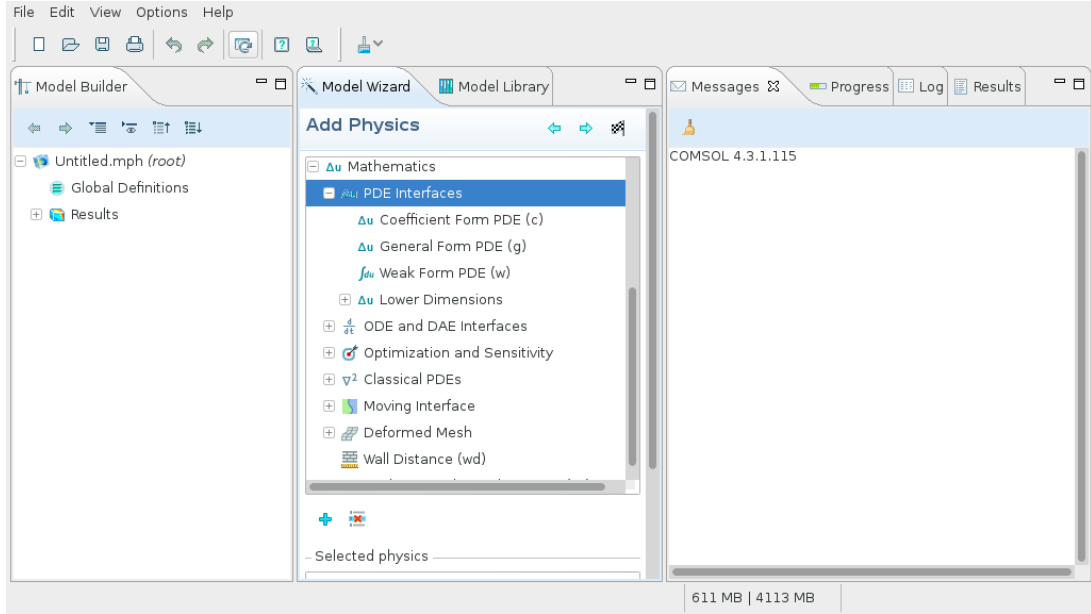


Figure 1: Mathematical interface.

## PDE Interface

*COMSOL Multiphysics* incorporates the coefficient, general and weak form of the PDE interfaces to formulate functional form of the PDE system. The coefficient and general form are used for continuous PDEs and they correspond completely to each other. The third form is a low-level method to formulate the problems in their integral shape, being closer to the discrete problem of the Finite Element Analysis. This form gives a broader freedom in expence of more complex utilization.

The ordinary technique in *COMSOL Multiphysics* to define a problem is via parameters presented in the physics interfaces like material properties or boundary conditions. Every interface formulates one or several PDEs with boundary conditions out of its parameters. Then *COMSOL Multiphysics* combines all equations and boundary conditions into a single system of the PDEs and boundary conditions, and solves it using its weak formulation.

It is impossible to predict all physical, chemical etc. problems and create corresponding physics interfaces. Therefore *COMSOL Multiphysics* offers a mathematical interface to define PDEs and other equations directly (Fig. Fig. 1).

## Coefficient Form

In the coefficient equation system the PDEs and boundary conditions are introduced in the following way:

$$\begin{cases} e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} + \nabla \cdot (-c \nabla u - \alpha u + \gamma) + \beta \cdot \nabla u + a u = f & \text{in } \Omega \\ \mathbf{n} \cdot (c \nabla u + \alpha u - \gamma) + q u = -g - h^T \mu & \text{on } \partial\Omega \\ h u = r & \text{on } \partial\Omega \end{cases} \quad (1)$$

- $\Omega$  is a computational domain or a union of such domains;

- $\partial\Omega$  is a domain boundary;
- $\mathbf{n}$  is an unit normal vector of the boundary  $\partial\Omega$  pointing out.

The first equation is a PDE which should be satisfied in the domain  $\Omega$ . The second and the third equations are boundary conditions what should be in effect on the border  $\partial\Omega$ . The second boundary condition is a generalization of the Neuman boundary condition, the third one is a general constraint, the Dirichlet boundary condition in the special case.

The Neumann boundary conditions are also known as the natural boundary conditions, since they do not appear explicitly in the weak form of the PDE. Corresponding boundary condition is called the flux or source in the PDE interface, since it determines the numerical value of the flux at the boundary.

The constraints and Dirichlet boundary conditions are known as the essential boundary conditions, since they introduce limitations to the space of the trial functions, which is not the part of the base equation. The PDE interface makes a difference between the Dirichlet boundary conditions and the constraints. The constraint defines that an arbitrary expression is zero at the boundary ( $R = 0$ ). At the same time the Dirichlet conditions is a special case specifying directly the value of the dependent variable at the boundary ( $u = r$ ).

Term  $-h^T \mu$  in the general Neumann conditions is a reaction term, forcing the constraint  $R = 0$ . If the reaction term is applied symmetrically to all dependent variables,

$$h = -\frac{dR}{du} \quad (2)$$

but other definitions are possible. Variable  $\mu$  is a Lagrange multiplier eliminated by solvers if they use standard constraints and therefore this multiplier usually does not appear directly in the equations.

The coefficients:

- $e_a$  - mass coefficient;
- $d_a$  - damping coefficient or mass coefficient;
- $c$  - diffusion coefficient;
- $\alpha$  - conservative flow convection coefficient;
- $\beta$  - convection coefficient;
- $a$  - absorption coefficient;
- $\gamma$  - conservative flow source;
- $f$  - source.

The coefficients  $c, \alpha, \gamma, \beta$ , and  $a$ , and terms  $f, g$ , and  $R$  or  $r$  may all be the functions of the spatial coordinates.

- The linearity condition of the PDE is ensured if the coefficients depend on the spatial coordinates only or remain constant.

- The PDE is nonlinear, if  $c, \alpha, \beta, a, h,$  or  $q$  depend on  $u$  or its derivatives (e.g. the components of  $\nabla u$ ) or if  $\gamma, f, g, R,$  or  $r$  are nonlinear in  $u$ .
- All coefficients are scalars in the general equation, except  $\alpha, \beta$  and  $\gamma,$  which can be a  $n$ -dimensional vectors. The coefficient  $c$  can be  $n$ -by- $n$  matrix to model anisotropic materials.

$e_a$  is a scalar or a matrix. If  $e_a$  equals to 0, the coefficient  $d_a$  is often called a mass coefficient.

## General Form

The system of time-dependent general form PDEs and boundary conditions can be expressed as following:

$$\begin{cases} e_a \ddot{u} + d_a \dot{u} + \nabla \Gamma = F & \text{in } \Omega \\ -\mathbf{n} \cdot \Gamma = G - h^T \mu & \text{on } \partial\Omega \\ 0 = R & \text{on } \partial\Omega \end{cases} \quad (3)$$

The first equation is the PDE. The second and the third equations are Neuman and Dirichlet boundary conditions correspondingly.

The coefficients:

- $e_a$  - mass coefficient;
- $d_a$  - damping coefficient or mass coefficient;
- $\Gamma$  - conservative flux vector;
- $f$  - source term.

The terms  $\Gamma, f, G,$  and  $R$  are the coefficients, what can depend on the spatial coordinates, solution  $u$  and its space derivatives. The coefficients  $f, G$  and  $R$  are scalars,  $\Gamma$  is the flux vector.  $-h^T \mu$  in the generalised Neumann conditions is a reaction term forcing the constraint  $R = 0$ . If the reaction term is applied symmetrically to all dependent variables, then

$$h = -\frac{dR}{du}$$

but other definitions are possible. The variable  $\mu$  is Lagrange's multiplier, which is eliminated by the solvers, if the standard conditions are used. Therefore this multiplier usually does not appear in the equations.

The relationship between the coefficient and general form is the following:

$$\begin{aligned} \Gamma &= -c \nabla u - \alpha u + \gamma \\ F &= f - \beta \nabla u - a u \\ G &= g \\ R &= r - u \end{aligned} \quad (4)$$

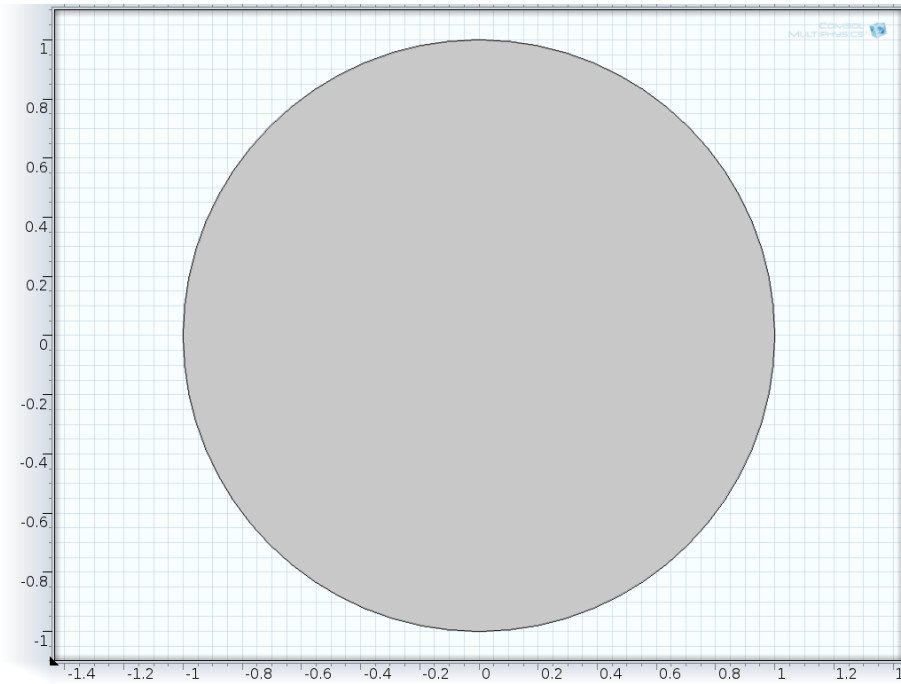


Figure 2: Unit circle.

## Poisson Equation

Present task solves the Poisson equation of the unit circle (Fig. 2). This is a classical PDE with an analytical solution. The full system of the PDEs of the unit circle  $\Omega$  is the following:

$$\begin{cases} -\nabla \cdot (\nabla u) = 1 & \text{in } \Omega \\ u = 0 & \text{on } \Omega \end{cases} \quad (5)$$

The analytical solution:

$$u(x, y) = \frac{1 - x^2 - y^2}{4} \quad (6)$$

---

**Task 1:**

- 1 Solve the PDE (5) in the coefficient form (1). Initial mesh should be user controllable mesh with *Predefined*  $\rightarrow$  *Normal* resolution.
  - 2 Present the values/equations of the coefficients and the boundary conditions.
  - 3 Plot the default solution.
  - 4 Plot the default solution with height information.
  - 5 Plot the function (6).
  - 6 Plot the solution and the function (6) at the same plot in the same limits. The solution plot type should be *wireframe*.
  - 7 Plot the maximum difference between the solution and the function (6) as a maximum error of the solution for each mesh resolution in the predefined range of *Extremely coarse* . . *Extremely fine* as a line plot. Which are the mesh parameters what affect the maximum error of the solution the most and how? The answer may include all necessary plots, tabels and data not listed here.
-

# Pellet

## Spherically Symmetrical Transport

Many of the models of the industrial-transport problems allow to make the assumption that the problem is spherical in its nature. This assumption has a great importance since this enables to eliminate two spatial coordinates resulting in 1D problem, which is computationally faster and requires lower amount on memory. Some applications what benefit from the spherical approach:

- the reaction and diffusion in the catalytic granules in the chemical reactors;
- the heat and mass transfer in the processing of the updated iron ore granules;
- any other process in the almost spherical drops.

The following prerequisites are needed to apply to ensure the validity of the spherical symmetry:

- the calculating domain has a spherical shape;
- the boundary condition of the external perimeter does not change upon the changing its location on the surface, i.e. no dependence on the spherical coordinates  $\theta$  and  $\phi$ ;
- in the case of the time-dependent problem, the material properties depend only on the radial distance to the center  $r$  and do not depend on the spherical coordinates  $\theta$  and  $\phi$ ;
- in the case of the time-dependent problem, the initial conditions depend only on the radial distance to the center  $r$  and do not depend on the spherical coordinates  $\theta$  and  $\phi$ .

## Model Definition

The following example simulates the short-term preheating of the granules of the loadstone ore. This is the first step to produce iron ore granules, what is a important raw material in the steel industry.

During the preheating of the loadstone ore granules the temperature remains in the range where the ignoring every phase change of the moisture is possible. Therefore one can use the short-term heat transfer equation with constant properties as spherically symmetrical. Also, the equations can be scaled to make the parametrisation of the radius more easier.

Although the real pellets are not ideally spherical, still the assumption of the spherical symmetry is used.

## Domain Equations

Starting with the time-dependent heat transfer equation:

$$\rho c_P \frac{\partial T}{\partial t} + \nabla \cdot (-k \nabla T) = Q \quad (7)$$

Transforming it into the polar coordinates:

$$\rho c_P \frac{\partial T}{\partial t} - k \left[ \frac{1}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial T}{\partial r} \right) + \frac{1}{r^2 \sin \theta} \frac{\partial}{\partial \theta} \left( \sin \theta \frac{\partial T}{\partial \theta} \right) + \frac{1}{r^2 \sin^2 \theta} \frac{\partial^2 T}{\partial \phi^2} \right] = Q \quad (8)$$



Figure 3: Area of change of r.

- $\rho$  - density ( $\text{kg/m}^3$ );
- $c_p$  - specific heat capacity ( $\text{J}/(\text{kg K})$ );
- $k$  - thermal conductivity ( $\text{W}/(\text{m K})$ );
- $Q$  - inner source ( $\text{W}/\text{m}^3$ );
- $r, \theta$  and  $\phi$  - spatial coordinates.

Assuming the perfect sphere with radius  $R_p$  and temperature to be independent on the spherical coordinates,  $\partial T/\partial\theta = \partial T/\partial\phi = 0$ , then

$$\rho c_P \frac{\partial T}{\partial t} + \frac{1}{r^2} \frac{\partial}{\partial r} (-kr^2 \frac{\partial T}{\partial r}) = Q \quad (9)$$

To avoid division by zero when  $r=0$ , multiply the latter equation by  $r^2$ :

$$r^2 \rho c_P \frac{\partial T}{\partial t} + \frac{\partial}{\partial r} (-kr^2 \frac{\partial T}{\partial r}) = r^2 Q \quad (10)$$

By introducing the dimensionless coordinate  $\hat{r}$  to scale the equation, an opportunity is taken to rapidly change the radius of the pellet without changing the size of the geometry or parametrization.

---

**Remark:** In *COMSOL Multiphysics* it is not necessary to use scaling variables since the solvers automatically scale the variables.

---

Replacing the dimensionless coordinate into the equation (10):

$$\hat{r} = \frac{r}{R_P} \quad \frac{\partial}{\partial r} = \frac{1}{R_P} \frac{\partial}{\partial \hat{r}} \quad (11)$$

It becomes:

$$\hat{r}^2 \rho c_P \frac{\partial T}{\partial t} + \frac{\partial}{\partial \hat{r}} \left( \frac{-k \hat{r}^2}{R_P^2} \frac{\partial T}{\partial \hat{r}} \right) = \hat{r}^2 Q \quad (12)$$

if  $r$  changes in the range of 0...1 (Fig. 3).

Similarly, the equations can be customise to describe flows in the porous environments, diffusion-reaction problems, etc.



## Boundary and Initial Conditions

Because of the symmetry of  $r=0$  the flux does not pass this point and therefore  $\partial T/\partial \hat{r} = 0$ .

At the surface of the pellet ( $r=1$ ) the thermal conductivity equation with the heat transfer coefficient  $h_s$  ( $W/(m^2 K)$ ) can be used for incoming heat flow:

$$q_{in} = h_s(T_{ext} - T) \quad (13)$$

This equations describes the gas flow with temperature  $T_{ext}$  around the pellet.  $T_{ext}$  equals to  $95^\circ C$ . Heat transfer coefficient  $h_s=1000 W/(m^2 K)$  and the initial temperature is  $25^\circ C$ .

## Modelling

Use 1D time-dependent PDE general form (3) to implement the Equation (12) and the boundary conditions.  $\hat{r}$  is the spatial coordinate in the model marked by  $rh$ . The correct equation should contain the folloing coefficients:

Coefficient	Equation
$e_a$	0
$d_a$	$\hat{r}^2 \rho c_P$
$\Gamma$ (flow vector)	$\frac{-k\hat{r}^2}{R_p^2} \frac{\partial T}{\partial \hat{r}}$
$F$ (source term)	$\hat{r}^2 Q_s$

One has to be careful when specifying the heat flux boundary conditions on the pellet surface. Since at the surface  $h_s(T_{ext} - T) = k\partial T/\partial r$ , the G has to be compensated accordingly:

$$G = \frac{\hat{r}^2}{R_p} h_s(T_{ext} - T) \quad (14)$$

In *COMSOL Multiphysics* notation  $G == g$ .

Boundary conditions:

Parameter	Boundary 1	Boundary 2
Boundary condition	Neumann	Neumann
G	0	$\frac{rh^2}{R_p} h_s(T_{ext} - T)$

Constants and parameters:

Name	Expression	Description
rho	2000	Density ( $kg/m^3$ )
cp	300	Heat capacity ( $J/(kg \cdot K)$ )
k	0.5	Thermal conductivity ( $W/(m \cdot K)$ )
Rp	0.005	Pellet radius (m)
Qs	0	Heat source ( $W/m^3$ )
hs	1000	Heat transfer coefficient ( $W/m^2 \cdot K$ )
Text	368	External temperature (K)
Tinit	298	Initial temperature (K)

The coordinate system has to be dimensionless. The dependent variable is  $T$ , the independent variables are  $rh$ ,  $theta$ ,  $phi$ . The shape function has to be *quadratic*.

---

**Remark:** The derivative function is  $d(F,x)$ , meaning the derivative of the variable  $F$  with respect to the variable  $x$ .

---

The mesh should be set to the predefined normal resolution of the general physics. Simulation time set in the range of 0..10 with the step of 0.25.

---

**Task 2:**

- 1 Solve the model.
  - 2 Plot the temperature vs. time in the center and at the edge of the pellet in the same plot.
  - 3 Is the simulation time long enough to enable the pellet to heat up through? What would be the appropriate actions? Apply them.
- 

---

**Task 3:**

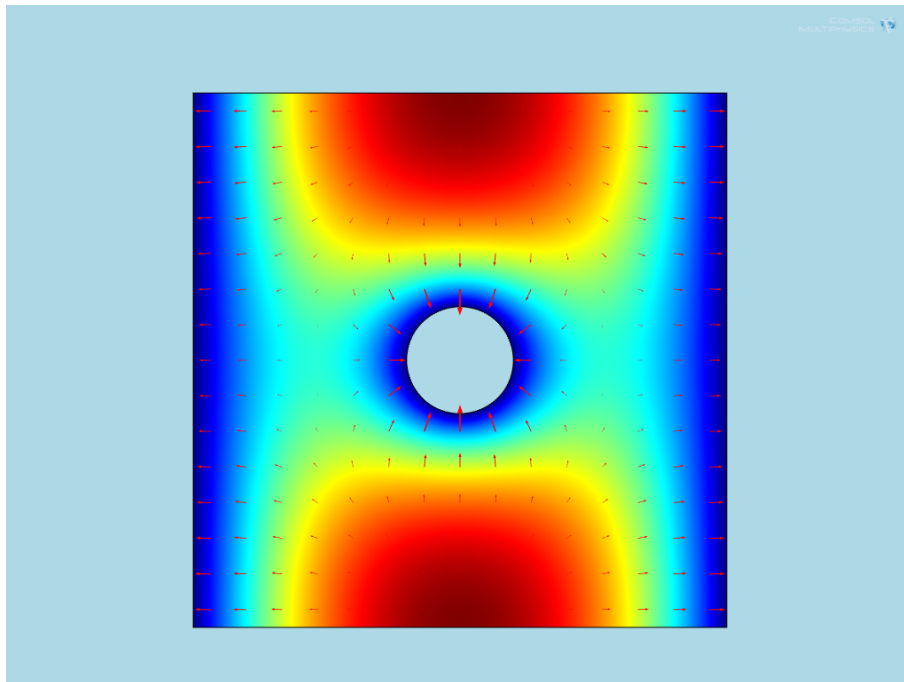
- 1 Add new time-dependent study to the model.
  - 2 Add the radius of the pellet  $R_p$  in the range of 0.001..0.01 m with the step of 0.001 m as a parameter to the model.
  - 3 Solve the model.
  - 4 Plot the temperature vs. time for different pellet radii separately for the center and the surface of the pellet.
  - 5 Plot the temperature stabilisation time vs. the pellet radius separately in the center and at the surface of the pellet. The temperature can be considered stable if its value is 99.5 % of the maximum temperature *Text*.
-

# Computational Physics II

## Practical Work

### Comsol Multiphysics

#### Resistive Heating



Heiki Kasemägi

October 11, 2013

# Contents

- Resistive Heating** . . . . . **2**
- Introduction . . . . . 2
- Passing the Practical Work . . . . . 2
- References . . . . . 2
- Resistive Heating . . . . . 3
- Definition of the Model . . . . . 3
- Tasks . . . . . 7

# Resistive Heating

## Introduction

The present Practical Work introduces the resistive heating aka Joule Heating. One has to create a computer model according to the description in this manual, solve the model and analyse it. The model is multiphysical, including the interactions between the electrical current and heat transfer. Two models have to be created:

- model based on the *COMSOL Multiphysics*'s module *Joule Heating*
- model based on the *COMSOL Multiphysics*'s PDE environment.

## Passing the Practical Work

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading “**Task #**”. The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may arise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size and legend(s). The report and accompanying files are submitted via moodle.

## References

The present practical work bases on the "Resistive Heating" example in “COMSOL Multiphysics Modeling Guide” Version 3.5.

## Resistive Heating

The core of the problem lies on the warming up of the copper plate due to the resistance to the electrical current passing the plate. The effect is also known as Joule Heating. This is a coupled interaction: the electrical resistivity of the material rises along with the temperature.

The generated resistive heat  $Q$  is proportional to the square of the magnitude of the electric current density  $J$ :

$$Q \propto |J|^2 \quad (1)$$

Current is proportional to the electric field strength  $E$ , which equals to the negative gradient of the potential  $V$ :

$$J = \sigma E \quad (2)$$

$$E = -\nabla V \quad (3)$$

The coefficient of the proportionality is the electric resistivity  $\rho = 1/\sigma$ , which is the reciprocal of the temperature-dependent electric conductivity  $\sigma = \sigma(T)$ . Combining it all together yields the complex equation:

$$Q = \frac{1}{\sigma} |J|^2 = \frac{1}{\sigma} |\sigma E|^2 = \sigma |\nabla V|^2 \quad (4)$$

The model should also take into account that the thermal conductivity depends on the temperature:

$$\sigma = \frac{\sigma_0}{1 + \alpha(T - T_0)} \quad (5)$$

$\sigma_0$  - conductivity at the reference temperature  $T_0$ ;  $\alpha$  - the temperature coefficient of the resistivity, which describes how the resistivity varies with temperature. The typical value for copper is 0.0039 1/K.

## Definition of the Model

The object of interest is a square copper plate of  $l$  m. There is a hole of  $0.2$  m diameter in the middle of the plate. The thickness of the plate is unimportant since the model is intended to solve in 2D. The opposite sides of the plate have a difference of the electric potential, all other sides are insulated, including the hole in the middle (Fig. 1). The potential difference induces the current that heats the plate.

In the conducting medium the potential is distributed for almost instantly, but the multiphysical model supposed to be dynamical since the heat transfer takes some time. Therefore, the multiphysical model consists of two parts: the heat transfer and conduction of the direct current. The coupling value is the resistive heat  $Q$  (4) created by the conduction of the direct current. Another coupling value is  $\sigma$  according to the Eq. (5).

The Partial Differential Equations (PDEs) and boundary conditions are presented in the coefficient equation system as following:

$$\begin{cases} e_a \frac{\partial^2 u}{\partial t^2} + d_a \frac{\partial u}{\partial t} + \nabla \cdot (-c \nabla u - \alpha u + \gamma) + \beta \cdot \nabla u + a u = f & \text{in } \Omega \\ \mathbf{n} \cdot (c \nabla u + \alpha u - \gamma) + q u = -g - h^T \mu & \text{on } \partial\Omega \\ h u = r & \text{on } \partial\Omega \end{cases} \quad (6)$$

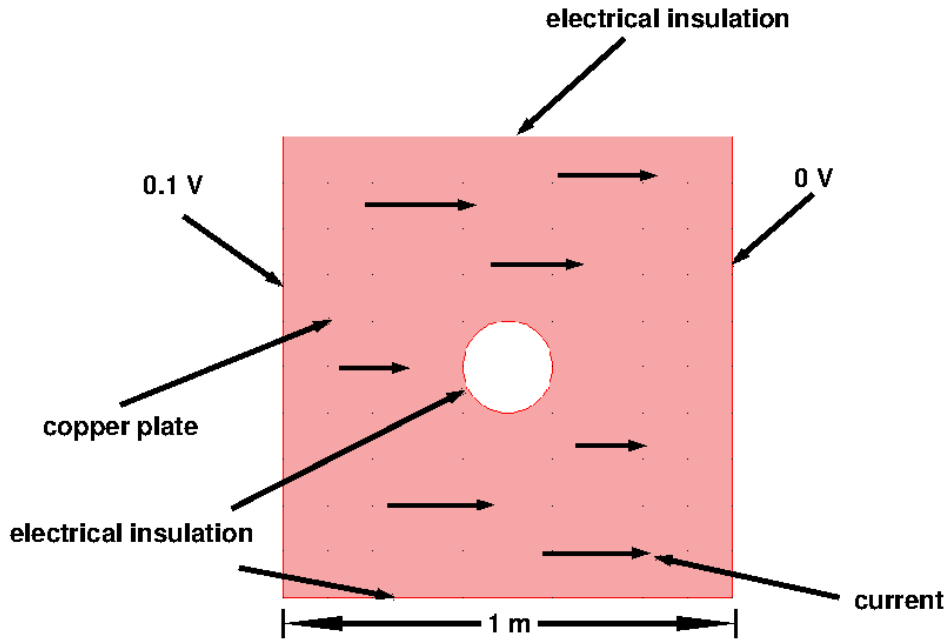


Figure 1: Geometry of the model and the electrical boundary conditions.

- $\Omega$  is a domain or the union of the domains.
- $\partial\Omega$  the domain boundary.
- $\mathbf{n}$  is a unit normal vector of the boundary  $\partial\Omega$  pointing out of the domain.

Coefficients:

- $e_a$  - mass coefficient;
- $d_a$  - damping or mass coefficient;
- $c$  - diffusion coefficient;
- $\alpha$  - convection coefficient of the conservative flux;
- $\beta$  - convection coefficient;
- $a$  - absorption coefficient;
- $\gamma$  - source of the conservative flux;
- $f$  - source.

Coefficients  $c, \alpha, \gamma, \beta, a$ , and members  $f, g$  and  $R$  or  $r$  may all be the functions of the spatial coordinates.

Heat transfer equation:

$$\rho C_p \frac{\partial T}{\partial t} + \nabla \cdot (-k \nabla T) = Q \quad (7)$$

- $\rho$  - density;
- $C_p$  - heat capacity at constant pressure;
- $k$  - thermal conductivity;
- $T$  - temperature;
- $Q$  - heat sources,  $[Q] = 1 \text{ W/m}^3$ ;
- $t$  - time.

In the case of stationary electric currents (charge relaxation time is much smaller than the external time) it has to take into account the stationary equation of continuity. In the stationary coordinate system the Ohm's Law is as following:

$$\mathbf{J} = \sigma \mathbf{E} + \mathbf{J}_e \quad (8)$$

- $\sigma$  - electrical conductivity;
- $\mathbf{J}$  - electric current density;
- $\mathbf{E}$  - electric field strength;
- $\mathbf{J}_e$  - externally generated current density.

The statical equation of continuity then obtains a shape:

$$\nabla \cdot \mathbf{J} = -\nabla \cdot (\sigma \nabla V - \mathbf{J}_e) = 0 \quad (9)$$

Considering the sources  $Q_j$  the equation transforms into:

$$-\nabla \cdot (\sigma \nabla V - \mathbf{J}_e) = Q_j \quad (10)$$

- $V$  - electric potential.

Parameters and constants required in the model:

Name	Value	Description
$r_0$	$1.754\text{E-}8 \text{ } \Omega \cdot m$	Resistance at the reference temperature
$T_0$	$20^\circ\text{C}$	Reference temperature
$\alpha$	$0.0039 \text{ } 1/\text{K}$	Temperature coefficient
$V_0$	$0.1 \text{ V}$	Electric potential
$\rho$	$8930 \text{ kg/m}^3$	Density
$C_p$	$340 \text{ J/(kg}\cdot\text{K)}$	Heat capacity
$k$	$384 \text{ W/(m}\cdot\text{K)}$	Isotropic thermal conductivity coefficient



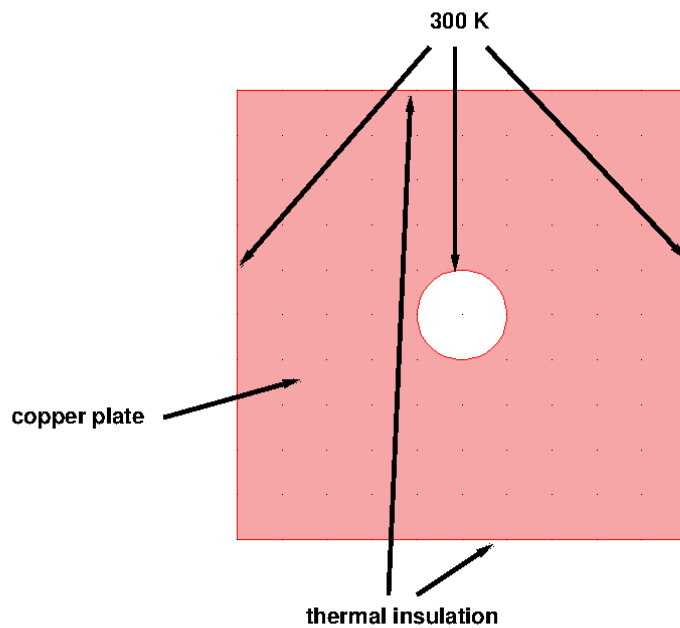


Figure 2: Model geometry and the thermal boundary conditions.

Thermal boundary conditions are introduced in Fig. 2. Basically, the constant temperature (300 K) air flow cools the plate on the sides and in the hole. The upper and lower boundary are thermally insulated.

The initial temperature of the plate is 300 K.

Electrical boundary conditions are presented in Fig. 1. The upper and lower boundary and the hole are electrically insulated.

The initial potential should be evenly distributed between the left boundary having a potential of 0.1 V and the right one with the potential of 0 V.

Mesh the plate with the default mesh.

# Tasks

---

**Task 1:** Model based on the *COMSOL Multiphysics Joule Heating* Module.

- 1 Solve the model during 2000 s with timestep of 50 s.
  - 2 Present the default plot.
  - 3 Complement the temperature plot with the total heat flux as arrow plot.
  - 4 Plot the temperature vs. simulation time at the point (0.5;0.75) assuming that the lower left corner has the coordinate (0;0). Plot also the point location.
- 

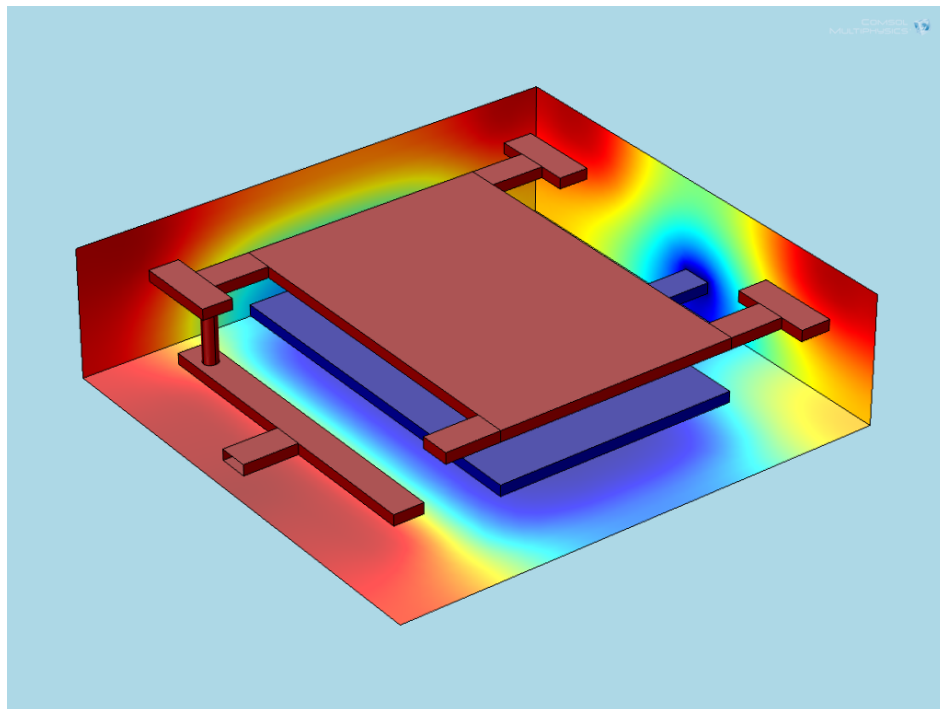
**Task 2:** Model based on the *COMSOL Multiphysics Coefficient Form PDE* Module.

- 1 Present the full description of the complete partial differential equations system in coefficient form describing the problem.
  - 2 Solve the model during 2000 s with timestep of 50 s.
  - 3 Present the default plot.
  - 4 Complement the temperature plot with the total heat flux as arrow plot.
  - 5 Plot the temperature vs. simulation time at the point (0.5;0.75) assuming that the lower left corner has the coordinate (0;0). Plot also the point location.
- 

**Remark:** All parameters needed to build up the model, including material description, are presented in this manual. Do not use the *COMSOL Multiphysics Material Library*.

---

Computational Physics II  
Practical Work  
Comsol Multiphysics  
Tunable MEMS Capacitor



Heiki Kasemägi

October 10, 2013

# Contents

<b>Tunable MEMS Capacitor</b>	<b>2</b>
Introduction . . . . .	2
Passing the Practical Work . . . . .	2
References . . . . .	2
Definition of the Model . . . . .	3
Geometry . . . . .	3
Definitions . . . . .	5
Material . . . . .	5
Electrostatics . . . . .	6
Tasks . . . . .	6

# Tunable MEMS Capacitor

## Introduction

It is possible to adjust the distance between the parallel capacitor's plates depending on the applied voltage. The plates are connected via a spring attached to one of the plates. Changing the spring length changes the distance between the plates. Knowing the characteristics of the spring and the voltage it is possible to calculate the distance between the plates and the capacity of the capacitor.

The capacitor presented in this model is a typical component in miscellaneous Micro-Electromechanical Systems (MEMS) operating in the electromagnetic fields in the frequency range of 300 MHz to 300 GHz.

## Passing the Practical Work

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading "**Task #**". The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may rise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size, title(s), legend(s) etc. The report and accompanying files are submitted via moodle.

## References

The present Practical Work bases on the "Tunable MEMS Capacitor" example in COMSOL Multiphysics Version 4.3.a Model Library.

## Definition of the Model

Electric scalar potential  $V$  satisfies the Poisson equation

$$-\nabla \cdot (\epsilon_0 \epsilon_r \nabla V) = \rho \quad (1)$$

- $\epsilon_0$  - absolute dielectric permittivity;
- $\epsilon_r$  - relative dielectric permittivity;
- $\rho$  - space charge density.

Electric field  $E$  and the displacement  $D$  are calculated from the gradient of  $V$ :

$$E = -\nabla V \quad (2)$$

$$D = \epsilon_0 \epsilon_r E \quad (3)$$

The capacity  $C$  of the parallel plate capacitor having charges  $+q$  and  $-q$  on the plates, the potential  $V$  between the plates and overlapping area  $A$  of the plates, is formulated as:

$$C = \frac{q}{V} = \epsilon_r \epsilon_0 \frac{A}{d} \quad (4)$$

Potential boundary conditions are applied to the plates and terminals of the capacitor. A port conditions keeps 1 V potential on the upper plate and connectors, the lower plate has ground potential. The surface of the surrounding box has a boundary condition with zero surface charge:

$$n \cdot D = 0 \quad (5)$$

## Geometry

The model consists of two plates. The lower one is a box having a narrow outlet on the bottom side to the backside of the box surrounding the plates. The upper plate has 3 T-connectors, two of them in the backside parallel to each other and T-roof parallel to the backside, anterior's T-roof is parallel to the frontside. There is a spring under the anterior T-connector represented by a simple cylinder in the model. The cylinder relies on the one side of the narrow connector parallel to the frontside. A outlet perpendicular to the connector connects its central bottom part to the frontside.

**Since all measures and coordinates are in micrometers, the model should be scaled accordingly to obtain the model sizes in meters.**

The upper plate of the capacitor consists of 10 blocks and a cylinder what should form a composite object without interior boundaries.

Draw the following boxes:

Name	Length			Base point of the axis		
	x	y	z	x	y	z
BLK1	22	60	8	0	240	46
BLK2	40	22	8	22	259	46
BLK3	176	262	8	62	19	46
BLK4	40	22	8	238	259	46
BLK5	22	60	8	278	240	46
BLK6	40	22	8	238	19	46
BLK7	22	60	8	278	0	46
BLK8	40	22	8	22	19	46
BLK9	22	229	8	0	41	0
BLK10	40	22	8	-40	139	0

Draw a cilinder:

- radius: 5.5;
- height: 38;
- x: 11;
- y: 250;
- z: 8.

The lower plate of the capacitor consists of following two blocks combined into a composite object without interior boundaries:

Name	Length			Base point of the axis		
	x	y	z	x	y	z
BLK1	176	226	8	62	19	8
BLK2	181	22	8	139	139	0

Surround these two plates of the capacitor by the block:

- width: 360;
- depth: 340;
- height: 94;
- x: -40;
- y: -20;
- z: -20.

## Definitions

*COMSOL Multiphysics* has a number of possibilities for different selections under *Model Builder* → *Model 1* → *Definitions: right button* → *Selection*:

- **Explicit** - normal selection tool to select individual geometric objects (e. g. edges);
- **Union** - the selection consists of the union of selections;
- **Intersection** - the selection is an intersection of selections;
- **Difference** - the selection is a difference between a set of one or more selections and another set of one or more selections;
- **Complement** - the selection is a complement (inverse) of one or more selections;
- **Adjacent** - the selection is an adjacent geometric object (e. g. edge) to one or more selections;
- **Ball** - the selection consists of geometric objects partially or completely inside the sphere;
- **Box** - the selection consists of geometric objects partially or completely inside the box;
- **Cylinder** - the selection consists of geometric objects partially or completely inside the cylinder.

It is useful to create the following selections:

1 electrode:

- create selection **Explicit**;
- rename it to “Electrode”;
- set parameters:
  - **Input Entities** → **Geometric entity level: Domain**; select domain 2 i.e. upper plate;
  - **Output Entities** → **Adjacent Boundaries**;

2 ground: the same selection scheme, select domain 3, i.e. lower plate;

3 dielectric: the only parameter to change is the domain, what should be set to 1, i.e. the shell surrounding the plates.

## Material

1 Create new material into the *Model Builder* → *Model 1* → *Materials* branch and rename it to **Dielectric**.

2 In the parameters window make domain level selection **Selection** → **Dielectric**.

3 Add an entry into the **Material Contents** section:



- Property: relative permittivity;
- Name: epsilon<sub>r</sub>;
- Value: 4.2.

## Electrostatics

- 1 *Electrostatics* → *Domain Selection* → *Dielectric*.
- 2 *Electrostatics: right button* → *Terminal* → *Boundary Selection* → *Electrode*.
- 3 *Terminal 1* → *Terminal type* → *Voltage*.
- 4 *Electrostatics: right button* → *Ground* → *Boundary Selection* → *Ground*.

## Tasks

---

### Task 1:

- 1 Solve the model using the default mesh.
- 2 Plot the default solution.
- 3 Plot the electric potential on the surface of the electrodes and the box surrounding them.

---

**Remark:** To remove the surfaces blocking the view:

- (a) *Results* → *Data Sets* → *Solution 1: right button* → *Add Selection* → *Geometric entity level: Boundary; Selection: All Boundaries*.
  - (b) Select only boundaries **3** and **5-78**. The easiest way is to select all boundaries and then remove **1, 2** and **4**.
- 

- 4 Calculate the capacity of the capacitor at least in two ways in picofarads:
    - (a) calculate the value of the built-in variable *C11*: *Results* → *Derived Values: param klahv* → *Global Evaluation* → *Replace Expression* → *Electrostatics* → *Terminals* → *Capacitance (es.C11)*;
    - (b) another way involves the equation (4).
-

---

**Task 2:**

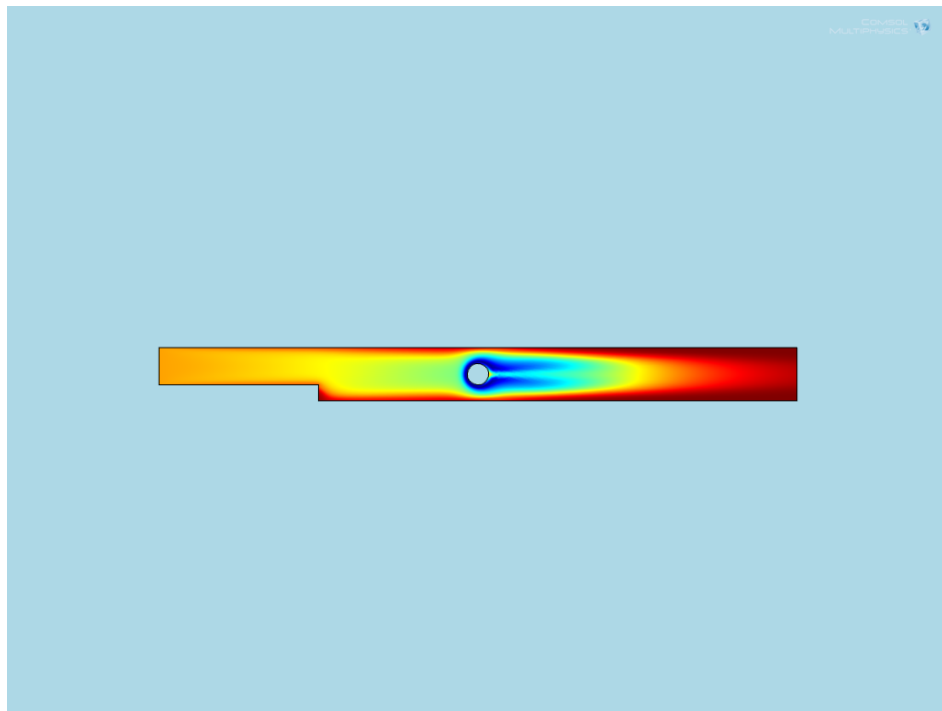
- 1 Turn the model into parametric one using the distance between the plates of the capacitor as a parameter. The lower plate can not be moved, also cannot be moved the connector of the upper plate under the cylindrical spring.
  - 2 Run the parametric simulations using the parameter values in the range of 25..45  $\mu\text{m}$  with 1  $\mu\text{m}$  step.
  - 3 Plot the capacitor's capacity as a function of the distance between the plates.
  - 4 Does the simulation result matches the theoretical one? Add the theoretical capacity as a function of the distance between the plates the the previous plot.
-

# Computational Physics II

## Practical Work

### Comsol Multiphysics

#### Thermal Decomposition in Parallel Plate Reactor



Heiki Kasemägi

October 11, 2013

# Contents

<b>Thermal Decomposition in Parallel Plate Reactor</b>	<b>2</b>
Introduction . . . . .	2
Passing the Practical Work . . . . .	2
References . . . . .	2
Definition of the Model . . . . .	3
Chemistry . . . . .	3
Momentum Transport . . . . .	4
Energy Transport . . . . .	5
Mass Transport . . . . .	6
Preparing the Model . . . . .	7
Modelling . . . . .	9
Geometry . . . . .	9
Laminar Flow . . . . .	9
Heat Transfer in Fluids . . . . .	11
Mass Transport and Chemical Reaction . . . . .	12

# Thermal Decomposition in Parallel Plate Reactor

## Introduction

The current practical work keeps its focus on the Chemical Engineering module and its potential to solve momentum, energy and mass transport related problems. The heat and mass transfer are coupled to the laminar flow to model the exothermic reactions taking place in the parallel plate reactor. This is an example of the *COMSOL Multiphysics* capabilities to systematically create and solve models with growing complexity using predefined modules.

## Passing the Practical Work

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading “**Task #**”. The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may rise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size, title(s), legend(s) etc. The report and accompanying files are submitted via moodle.

## References

The present practical work bases on the "Thermal Decomposition" example in the Model Library of COMSOL Multiphysics Version 4.3.a.

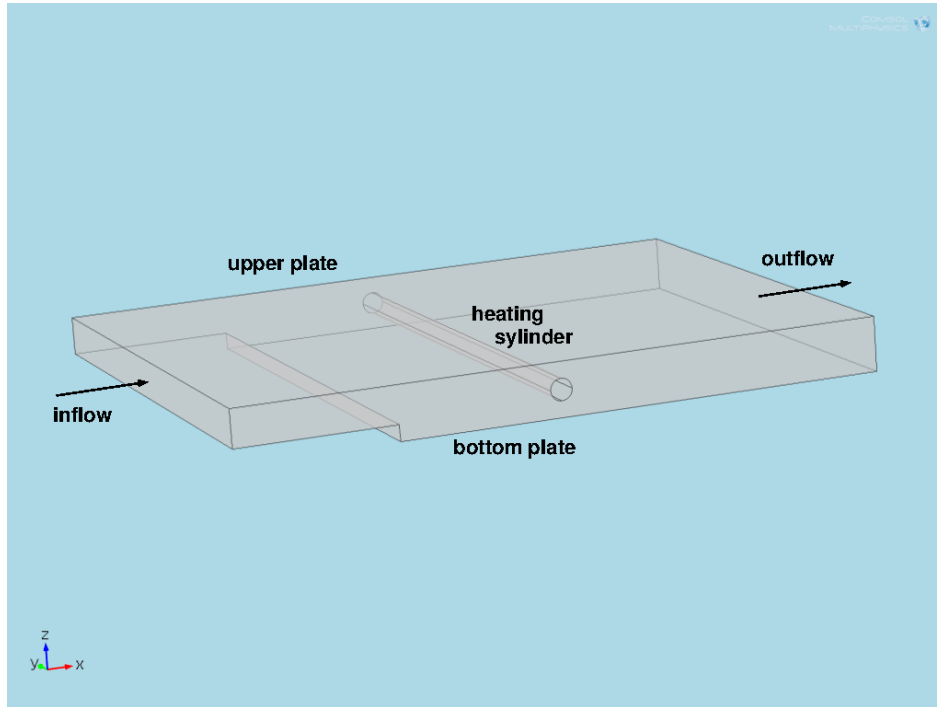


Figure 1: Parallel plate reactor.

## Definition of the Model

Present model studies the unimolecular decomposition of the chemical passing through the parallel plate reactor (Fig. 1). A heat sensitive compound is present in a water solution. After entering into the reactor the fluid first experiences the expansion due to the step in the bottom plate. On exiting the fluid passes the heating cylinder.

The width of the short entrance area of the reactor is considerably large than its height. Therefore it is reasonable to assume the parabolic velocity profile between the plates. At the same time the velocity between the side walls should remain almost constant. This gives a chance to reduce the 3D problem into 2D without any significant impact on the validity of the model (Fig. 2).

## Chemistry

A heat sensitive chemical (A) decomposes into the fragments (F) according to the unimolecular reaction



Reaction rate (mol/(m<sup>3</sup>·s)):

$$rate = kc_A \quad (2)$$

where the rate constant  $k$  (1/s) depends on the temperature according to the Arrhenius equation

$$k = A \exp\left(-\frac{E}{RT}\right) \quad (3)$$

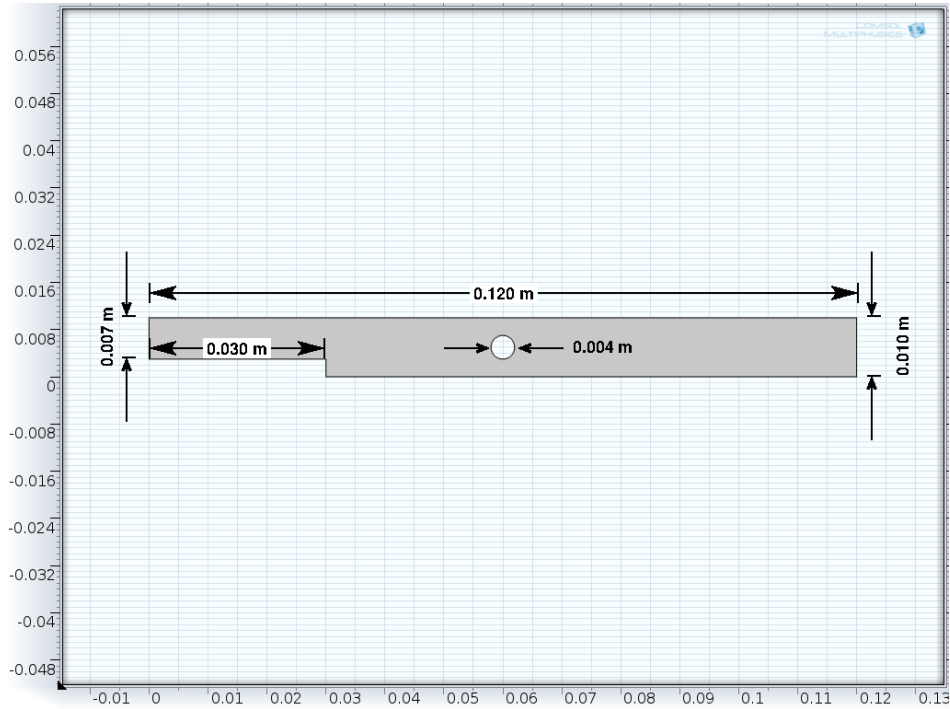


Figure 2: 2D geometry of the model.

- $A$  - frequency factor (1E10 1/s);
- $E$  - activation energy (7.2E3 J/mol);
- $R$  - universal gas constant (8.314 J/(mol K));
- $T$  - temperature (K).

In addition, the decomposition reaction is exothermic, and the rate of energy expelled is

$$Q = rate \cdot H \quad (4)$$

$H$  - heat of reaction (100 kJ/mol).

The conversion of species  $A$  in the reactor is dependent on the residence time, i. e. the details of the fluid flow. Also, the temperature distribution has its impact on the decomposition.

## Momentum Transport

The Navier-Stokes equations, what represent the momentum transport and are solved by default in the single phase flow interface, are the compressible formulation of the continuity:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0 \quad (5)$$

and the momentum equations

$$\rho \frac{\partial \mathbf{u}}{\partial t} + \rho \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla p + \nabla \cdot (\mu(\nabla \mathbf{u} + (\nabla \mathbf{u})^T) - \frac{2}{3}\mu(\nabla \cdot \mathbf{u})\mathbf{I}) + \mathbf{F} \quad (6)$$

- $\eta$  - dynamic viscosity (Ns/m<sup>2</sup>);
- $\mathbf{u}$  - velocity (m/s);
- $\rho$  - fluid density (kg/m<sup>3</sup>);
- $p$  - pressure (Pa);
- $\mathbf{F}$  - body force term (N/m<sup>3</sup>).

In the present model, the steady-state problem is solved and the first term in the Equation (5) and (6) can be cancelled. The Equation (5) describes the flow of a Newtonian fluid in the laminar flow regime.

Besides the domain equations, the proper boundary conditions have to be selected. The velocity vector normal to the boundary specified in the inlet is

$$\mathbf{u} \cdot \mathbf{n} = u_0 \quad (7)$$

Pressure specified in the outlet is

$$p = p_0 \quad (8)$$

Finally, the *no-slip* boundary conditions specifies zero flow velocity on the surfaces of the plates and heating cylinder:

$$\mathbf{u} = 0 \quad (9)$$

By selecting the **Laminar Flow** module it is easy to associate the balance of the momentum (5) and boundary conditions (7), (8) and (9).

## Energy Transport

Energy balance equation applied to the reactor domain considers the heat transfer via convection and conduction:

$$\nabla \cdot (-k\nabla T) + \rho C_p(\mathbf{u} \cdot \nabla)T = Q \quad (10)$$

- $C_p$  - specific heat capacity (J/(kg K));
- $k$  - thermal conductivity (W/(m K));
- $Q$  - sink or source term (W/m<sup>3</sup>).



The boundary condition in the inlet and on the surface of the heating silinder is the temperature:

$$T = T_0 \quad (11)$$

$$T = T_{cyl} \quad (12)$$

The outlet boundary condition is *Outflow*. This sets that all the energy passes the outlet boundary by means of convection. At the same time it means that the conducting heat flux through this boundary is 0:

$$\mathbf{q}_{cond} \cdot \mathbf{n} = -k\nabla T \cdot \mathbf{n} = 0 \quad (13)$$

and the equation of the total flux becomes

$$\mathbf{q} \cdot \mathbf{n} = \rho C_p T \mathbf{u} \cdot \mathbf{n} \quad (14)$$

This is a useful boundary condition, especially in the energy balance situation where the convection is dominating and the output temperature is unknown.

Finally assume no energy transfer through the plates of the reactor, that is a thermal insulation boundary condition:

$$\mathbf{q} \cdot \mathbf{n} = 0 \quad (15)$$

Energy balance (10) and boundary conditions (11)..(15) can be added to the model via *Heat Transfer* physics interface.

## Mass Transport

The mass transfer in the reactor domain is determined by stationary convection and diffusion equation:

$$\nabla \cdot (-D_i \nabla c_i) + \mathbf{u} \cdot \nabla c_i = R_i \quad (16)$$

- $D_i$  - diffusion coefficient (m<sup>2</sup>/s);
- $R_i$  - reaction term (mol/(m<sup>3</sup> s)).

Equation (16) assumes the species are fully diluted in a solvent.

The concentration of species A at the inlet becomes a boundary condition:

$$c_i = c_{i,0} \quad (17)$$

Convection driven mass flow through the boundary becomes an outlet boundary condition. This assumes that any mass flux due to diffusion is zero

$$\mathbf{n} \cdot (-D_i \nabla c_i) = 0 \quad (18)$$

so that

$$\mathbf{N}_i \cdot \mathbf{n} = c_i \mathbf{u} \cdot \mathbf{n} \quad (19)$$

Finally assume no mass transport at the surfaces of the plates and the cylinder across the boundaries, that is, an insulation boundary condition:

$$\mathbf{N}_i \cdot \mathbf{n} = 0 \quad (20)$$

*Diluted Species* module adds mass balance (16) and boundary conditions (17)..(20) to the model.

## Preparing the Model

It is necessary to collect the data describing the reacting flow before starting the modelling. For example, the flow modelling depends on the fluid density and viscosity. Mass transport needs the diffusion coefficients and the reaction kinetics.

Another part of the preparations involves selecting the appropriate equations and physics interfaces and investigating the couplings between different transport equations.

### Transport Properties

The term transport properties refers to the physical properties occurring in the transport equations. Momentum and heat transfer equations (5) and (10) require following fluid-specific transport properties:

- viscosity ( $\eta$ );
- density ( $\rho$ );
- thermal conductivity ( $k$ );
- heat capacity ( $C_p$ ).

Mass transport equation (16) requires the diffusivities ( $D_i$ ) as the species-specific properties.

Accurate simulation results depend on the appropriate values of the transport properties. In the present model, the water temperature with the dissolved compound A is 300 K in the inlet. Since the water is the solvent, it can assume that its physical properties are the same for the entire fluid. The hottest spot of the reactor is at 325 K. Table 1 lists the transport properties of the water along with diffusivities of A in the water at 300 K and 325 K.

Initially the constant values of the transport properties may be used. Later on, it is possible to load

Property	300 K	325 K
Density ( $\text{kg/m}^3$ )	997	987
Viscosity ( $\text{Ns/m}^2$ )	8.5E-4	5.3E-4
Thermal conductivity ( $\text{W/(m K)}$ )	0.62	0.66
Heat capacity ( $\text{J/(kg K)}$ )	4180	4182
Diffusivity ( $\text{m}^2/\text{s}$ )	2.0E-9	2.0E-9

Table 1: Transport properties of water.

the temperature-dependent values from the *COMSOL Multiphysics* Material Library. This ensures the model accuracy along with the coupling level of the equations system.

## The Flow Regime

The Reynolds number determines if the nature of the flow is laminar or turbulent:

$$Re = \frac{\rho u d}{\eta} \quad (21)$$

As a rule of thumb, the Reynolds number in the range of 2000 to 2500 shows the transitions from stable streamlines to stable turbulent flow. It is always wise to estimate the Reynolds number of the flow in the model since its magnitude shows if the flow regime used in the model is correct.

In this case the Reynolds number can be evaluated based on the values in the previous table, the stream velocity of  $5E-4$  m/s and the characteristic length of  $=0.007$  m:

$$Re = \frac{997 \cdot 5E-4 \cdot 0.007}{8.5E-4} \approx 4 \quad (22)$$

The Reynolds number for 325 K has almost the same value.

Therefore, the Reynolds numbers are well in the limits of the laminar flow regime.

## Dilute or Concentrated Mixtures

While modelling the mass transport, it is useful to distinguish the dilute and concentrated mixtures. Fick's law adequately describes the diffusional transport of dilute mixtures assuming that the fluid transport properties are the same as for the solvent. For concentrated mixtures, the mass transport equations should include, for example, Maxwell-Stefan diffusion. Also, the transport properties of the fluid then depend on the components of the solution.

Suitable module for the dilute mixtures in *COMSOL Multiphysics* is *Transport of Diluted Species*, the *Transport of Concentrated Species* module is good for the concentrated mixtures.

Generally, the concentrations upto 10 mole% are considered as dilute mixture. In the present case, the concentration of species A in the water is  $1000 \text{ mol/m}^3$ . The concentration of the pure water is  $55500 \text{ mol/m}^3$ , therefore the molar fraction of A is about 2 %. Since it is a dilute solution, the *Diluted Species* meets the model conditions and the water transport properties represent the corresponding properties of the solution.

## Solving Coupled Models

As mentioned earlier, the chemical process taking place in the reactor depends both on the fluid flow and the temperature distribution in the reactor. More specifically, the mass transport equation

$$\frac{\partial c_i}{\partial t} + \nabla \cdot (-D_i \nabla c_i + c_i \mathbf{u}) = R_i \quad (23)$$

depends on the velocity vector  $\mathbf{u}$ , which is a solution of the momentum transport equation (6).

The source term  $R_i$  in the equation (23) is a function of the temperature. The temperature, at the other hand, is a dependent variable in the energy transport equation:

$$\rho C_p \frac{\partial T}{\partial t} + \nabla \cdot (-k \nabla T) + \rho C_p \mathbf{u} \cdot \nabla T = Q \quad (24)$$

The reasonable approach to solve this kind of coupled equation system is to take it step-by-step, analysing the relations separately and one after another.

In this case, at first the reaction heat  $Q$  is ignored, obtaining a weak two-way coupling between the transport equations:

- the momentum transport is weakly coupled to the energy and mass transport via material properties;
- the energy transport depends only on the momentum transport;
- the mass transport depends both on the momentum and energy transport.

Thus, the problem should be solvable gradually in the following order: first solve the momentum transport + energy transport, then add the mass transport and investigate the differences.

The result of the last step is a fully coupled system:

- the transport of momentum depends on the energy transport;
- the transport of energy depends on the momentum and mass transport (heat of the reaction is added);
- the mass transport depends both on the momentum and energy transport.

In this case, all equations describing the transport phenomena are solved simultaneously. Sometimes the solver needs a good initial solution obtainable from the stepwise approach where the couplings are added one-by-one on behalf of the more accurate results.

## Modelling

### Geometry

The 2D geometry of the reactor is in Fig. 2 and the dimensions are in the Table 2.

### Laminar Flow

- Add the water as a material from the Material Library.
- Inlet: velocity  $U_0=5E-4$  m/s.
- Outlet: pressure  $p_0=0$ .

h

Name	Value	Description
H1	1 cm	Reactor height
W1	12 cm	Reactor length
H2	3 mm	Step height
W2	3 cm	Step length
R1	2 mm	Cylinder radius
SX	6 cm	Cylinder centre distance from the inlet
SY	5 mm	Cylinder centre distance from the bottom plate

Table 2: Model dimensions.

---

**Task 1:**

- 1 Solve the Laminar Flow case.
  - 2 Save the solution for subsequent simulations: *Model Builder* → *Study 1* → *Solver Configurations* → *Solver 1: right button* → *Solution* → *Copy* → *Copy 2: right button* → *Rename: isothermal flow*.
  - 3 Add the arrow surface to the velocity plot.
  - 4 The cross-section area of the fluid increases after the step and decreases in alignment to the cylinder. How this affects the velocity? Illustrate it with the line plot along the line parallel to the plates connecting the inlet and outlet through the cylinder centre.
-

## Heat Transfer in Fluids

---

### Task 2:

- 1 Add the Heat Transfer in Fluids to the model.
- 2 Couple the heat transfer and laminar flow.
- 3 Inlet temperature is 300 K.
- 4 The temperature of the heating cylinder is 325 K.
- 5 Couple the laminar flow to the heat transfer.
- 6 Use the solution of the isothermal flow as an initial guess: *Model Builder* → *Study 1* → *Solver Configurations* → *Solver 1* → *Dependent Variables 1* →:
  - *General* → *Defined by study step* = *User defined*;
  - *Initial Values of Variables Solved For* → *Method* = *Solution*;
  - *Initial Values of Variables Solved For* → *Solution* = *isothermal flow*.
- 7 Solve the model.
- 8 Copy and rename the solution as *nonisothermal flow*.
- 9 Plot the temperature as 2D plot.
- 10 Plot the temperature as a line plot along the line parallel to the plates connecting the inlet and outlet through the cylinder centre.

---

**Remark:** *Data set* list enables to make a selection among the stored solutions. *Solution 1* is the current solution. *Solution2* corresponds to the stored isothermal flow case and *Solution 3* non-isothermal flow case.

---

## Mass Transport and Chemical Reaction

---

### Task 3:

- 1 Add the Diluted Species Transport to the model.
  - 2 Set  $c_A$  as a dependent variable.
  - 3 Couple the fluid flow to the transport of species.
  - 4  $D_{cA} = 2e-9$ .
  - 5 Input concentration:  $c_{0,cA} = 1000$ .
  - 6 Insert the table of the variables (Table 3).
  - 7 Add reaction rate:  $R_{cA} = -rate$ .
  - 8 Since the chemical reactions are exothermic, add the heat source  $Q = H*rate$  to the heat transfer.
  - 9 Solve the model with the solution of the non-isothermal solution as an initial guess.
  - 10 Plot temperature in 2D.
  - 11 Plot the concentration of species A as 2D plot.
  - 12 How the temperature distribution changes in the reactor if the reaction rate is taken into account? Illustrate this with the line plot of the temperature of the non-isothermal flow and final temperature (on the same plot) along the line parallel to the plates connecting the inlet and outlet through the cylinder centre.
  - 13 Where is the reaction centre?
  - 14 Why is the decomposition higher in the inlet part of the reactor near the walls and after the step where the temperature is relatively low?
  - 15 Why is the concentration of the species relatively higher in the region after the heating cylinder?
-

Name	Expression	Description
E	72[kJ/mol]	Activation energy
H	100[kJ/mol]	Reaction heat
A	1E10[1/s]	Frequency factor
k	$A \cdot \exp(-E/(8.134[l/(\text{mol} \cdot \text{K})] \cdot T))$	Rate factor
rate	$k \cdot c_A$	Reaction rate

Table 3: Parameters of the reactive transport.

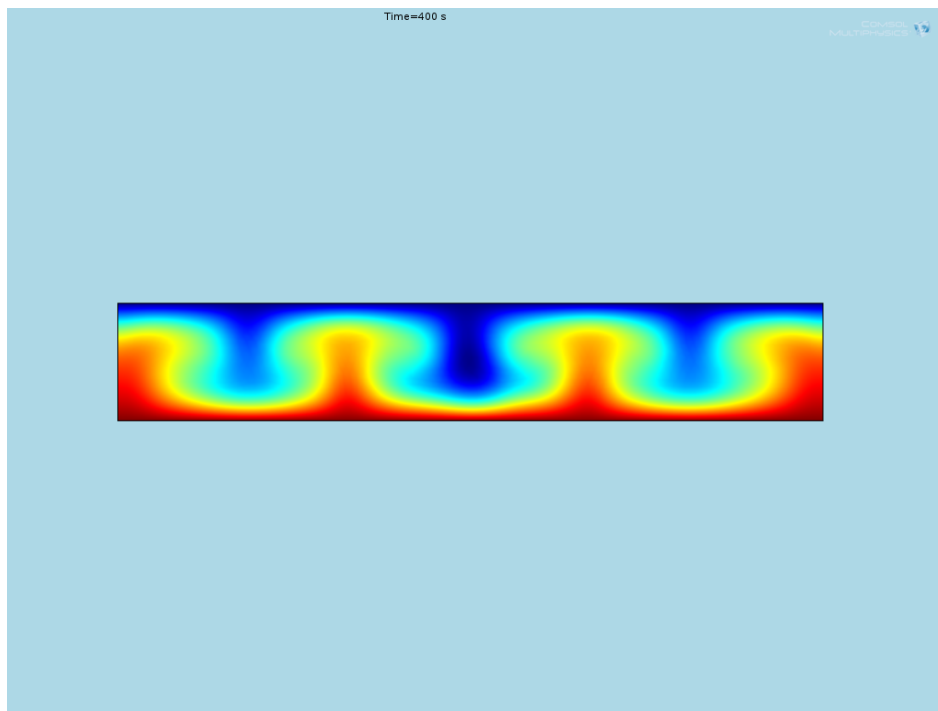


# Computational Physics II

## Practical Work

### Comsol Multiphysics

#### Beam, Plate, Rayleigh-Bernard Instability



Heiki Kasemägi

October 10, 2013

# Contents

<b>Passing the Practical Work</b>	<b>2</b>
References . . . . .	2
<b>Eigenvalue Analysis of an Elastic Beam</b>	<b>3</b>
<b>Elastic Linear Plate</b>	<b>4</b>
<b>Transient Flow and Heat Equations - Rayleigh-Benard Instability</b>	<b>5</b>

# Passing the Practical Work

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading “**Task #**”. The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may rise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size, title(s), legend(s) etc. The report and accompanying files are submitted via moodle.

## References

“Elmer GUI Tutorials”, Peter Råback,  
<http://www.nic.funet.fi/pub/sci/physics/elmer/doc/ElmerTutorials.pdf>

# Eigenvalue Analysis of an Elastic Beam

The object of interest is a homogeneous elastic silicon beam:

- length: 1 m;
- height: 0.1 m;
- width: 0.2 m;
- density  $\rho$ : 2330 kg/m<sup>3</sup>;
- Poisson ratio: 0.3;
- Young modulus: 1E11 N/m<sup>2</sup>.

The beam is supported in both ends. The problem is to calculate the eigenvalues of the beam. Mathematical equation to solve is

$$-\rho\omega^2\phi = \nabla \cdot \tau(\phi) \quad (1)$$

- $\omega^2$  - eigenvalue;
- $\omega$  - angular frequency;
- $\phi$  - corresponding vibration mode;
- $\tau$  - stress tensor.

---

## Task 1:

- 1 Create the model of the described elastic beam and solve its eigenproblem.
  - 2 Plot the first 6 eigenfrequencies as surface displacement plots.
  - 3 Plot the total displacement of the beam center as a function of the eigenfrequencies for the first 6 eigenfrequencies.
-

# Elastic Linear Plate

The object of interest is a L-shaped iron plate under the load. The plate properties:

- longer side length: 2 m;
- shorter side length: 1 m;
- the surface area of the plate: 3 m<sup>2</sup>;
- plate initial thickness: 1 cm;
- the load: 15300 kg;
- all the edges are fastened and not deflecting nor rotating;
- material: iron (pick from the *COMSOL Multiphysics Material Library*).

The load is the sand with total mass of 15300 distributed uniformly over the plate. The task is to calculate the deflection of the plate under the applied load.

---

## Task 2:

- 1 Create the geometry of the plate, mesh it, determine domain and boundary conditions, solve the problem.
  - 2 Plot the von Misses stress along with the deformation.
  - 3 Solve the model for different plate thickness in the range of 1 . . 10 cm with 1 cm step.
  - 4 Plot the deformation as a function of the plate thickness along the line connecting the opposite inflection points on the upper side of the plate.
  - 5 Plot the maximum deflection as a function of the plate thickness.
-

# Transient Flow and Heat Equations - Rayleigh-Benard Instability

Rayleigh-Benard convection is one of the subtypes of the natural convection, revealing itself for example if the fluid layer between two parallel plates is heated from below bringing up the regular structure of convection or Bernard cells.

Gravity of buoyancy is responsible for this phenomenon. The primary movement is the upward flow of the fluid with smaller density from the heated bottom layer. This flow organises spontaneously into a ordered cell structure. Since there is the density gradient between the bottom and upper layer, then the gravity tries to push the cooler and more dense fluid downwards. The viscosity counteracts to this process. The cells appear at the critical moment.

The task is to simulate the Rayleigh-Benard instability in the qdrilateral 2D domain with the width of 0.06 m and height of 0.01 m. The fluid is water. Material parameters are listed in the Table 1. There is a temperature difference of 0.5 K between the bottom and top boundary. The temperature

Parameter	Value
Reference density	1000 kg/m <sup>3</sup>
Viscosity	1040E-6 Ns/m <sup>2</sup>
Heat capacity	4190 J/(kg·K)
Heat conductivity	0.6 W/(m·K)
Heat expansion coefficient	1.8E-4 K <sup>-1</sup>

Table 1: Material parameters.

of the bottom boundary is 283.5 K and the top boundary 283 K. The initial temperature of the domain is 283 K. There is no flow through any boundary and the shorter boundaries are thermally insulated. Use quadratic discretization of the temperature, for the laminar flow of the fluid use **P2 + P1** scheme. The fluid is incompressible.

The density of the water is inversely proportional to the temperature. Heated water starts to flow upwards and cooler water downwards due to the gravity. At the present case there is an assumption that the Boussinesq approximation is in effect for the flow of the thermal uncompressed fluid. In other words, the density of the uncompressed Navier-Stokes equation can be defined through the

Boussinesq approximation:

$$\rho = \rho_0(1 - \beta(T - T_0))$$

where  $\beta$  is the heat expansion coefficient and subscript 0 marks the reference state.

The simulation time in the 2D case should be at least 400 s with 1 s step.

---

**Task 3:**

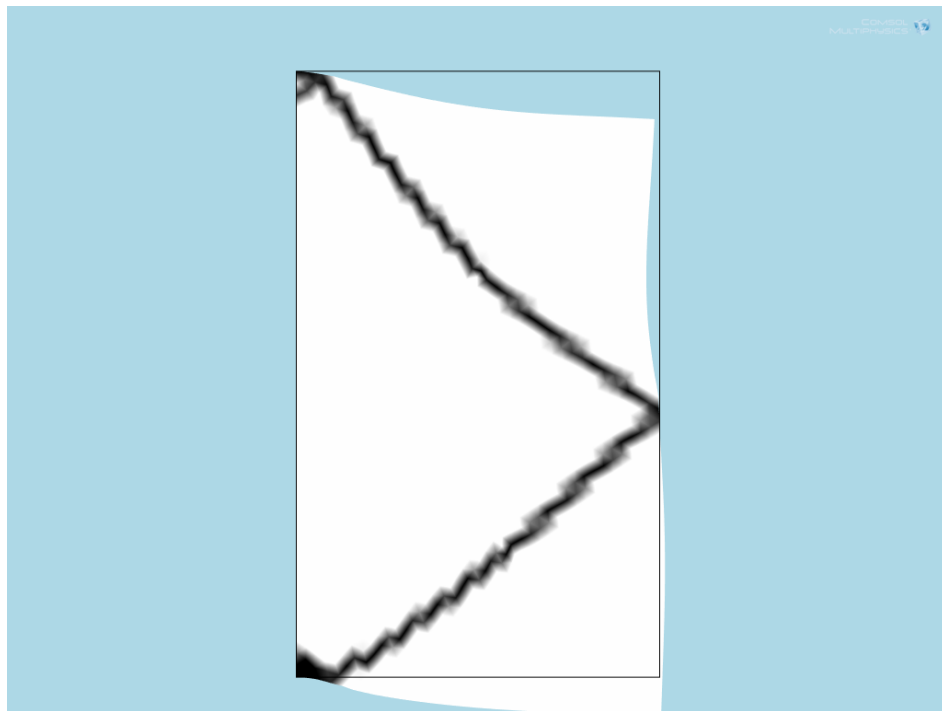
- 1 Create the model of the described problem and solve it.
  - 2 Make an animation of the evolution of the temperature and fluid velocity all over the simulation time in the same animation. The fluid velocity is represented by the arrows.
-

# Computational Physics II

## Practical Work

### Comsol Multiphysics

#### Optimization



Heiki Kasemägi

October 10, 2013



# Contents

<b>Optimization</b>	<b>2</b>
Introduction . . . . .	2
Passing the Practical Work . . . . .	2
References . . . . .	3
Approximation of Data . . . . .	4
Selecting the Module . . . . .	4
Geometry . . . . .	4
Definitions . . . . .	4
Optimisation . . . . .	5
Mesh . . . . .	5
Simulation . . . . .	5
Topology Optimization . . . . .	8
Defining the Optimization Problem . . . . .	8
Modelling . . . . .	8
Bracket . . . . .	12

# Optimization

## Introduction

The term optimization can be defined as “the design and operation of a system or process to make it as good as possible in some defined sense” [wiktionary].

The problem setup consists of three parts:

**objective function:** the features describing the model;

**design variable(s):** the inputs of the model that should be improved or changed to get the desired result;

**limiting constraint(s):** constraints to the design variables or operating conditions to satisfy.

The optimization improves the model by changing the design variables, while satisfying the constraints. Any of the model inputs like geometric dimensions, part shapes, material properties, material distribution can be in the rôle of the design variable. Any of the model outputs can be used to define the objective function.

## Passing the Practical Work

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading “**Task #**”. The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may arise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size, title(s), legend(s) etc. The report and accompanying files are submitted via moodle.

## References

The present practical work bases on the “COMSOL Multiphysics. Introduction to Optimization Module. Ver. 4.3.a”, which is a part of the COMSOL Multiphysics ver. 4.3.a distribution.

COMSOL Multiphysics Optimization Module overview, <http://www.comsol.com/optimization-module>

## Approximation of Data

The Optimization Module is used to approximate the experimental data in the present example. The aim is to find the values of two parameters what define the model of the non-linear material based on the experimental data. Materials model is hyper-elastic Mooney-Rivlin material. The parameters defining the model are  $C_{01}$  and  $C_{10}$ .

### Selecting the Module

- 1 *Model Wizard* → *1D* → *Next*.
- 2 *Add Physics* → *Mathematics* → *Optimization And Sensitivity* → *Optimization (opt): right button* → *Add selected* → *Next*.
- 3 *Preset Studies* → *Stationary* → *Finish*.

### Geometry

The 1D geometry represents the range of the strain: 0..0.37.

- 1 *Model Builder* → *Geometry 1: right button* → *Interval*.
- 2 *Right endpoint: 0.37*.
- 3 Click the *Build All* button.

### Definitions

- 1 Define interpolation function: *Model Builder* → *Model 1: right button* → *Functions* → *Definitions*.
- 2 Rename the function to *RawData*. This interpolation function uses measured stress-strain data as a source for curve fitting.
- 3 Insert data according to the Table 1.
- 4 *Model Builder* → *Definitions* → *Model Couplings* → *Integration* → *Source Selection* → *Selection* → *All domains*.
- 5 Define the variables  $L$  and  $P$  what correspond to the parameters  $lambda$  and  $P$  accordingly in the model simulating the stress in the Mooney-Rivlin hyperelastic material. Also define the least squares error as a difference between the model and experimental data. The variables are defined in the Table 2. One has not to be worried about the message that  $C_{10}$  and  $C_{01}$  are undefined at the moment 'cause its true.

t	f(t)
0	0
0.075	4.9e5
0.103	8.67e5
0.15	1.36e6
0.174	1.55e6
0.2004	1.71e6
0.25	1.95e6
0.305	2.10e6
0.351	2.17e6
0.37	2.21e6

Table 1: Initial data.

Name	Expression	Description
L	$1+x[1/m]$	Mooney-Rivlini parameter, lambda
P	$2*(L-1/L^2)*(C10+C01/L)$	Mooney-Rivlini parameter, P
SQDiff	$\text{intop1}((\text{RawData}(x[1/m])-\text{P})^2)$	Least-squares error

Table 2: Variables.

## Optimisation

The global goal is to minimize the least-squares error between the model and experimental data by changing the values of the two model parameters,  $C_{01}$  and  $C_{10}$ .

**1 Model Builder → Model 1 → Optimization: right button → Global Objective → Objective expression: SQDiff.**

**2 Model Builder → Model 1 → Optimization: right button → Global Control Variables** and insert the following variables with defined initial values:

- C01: 0;
- C10: 0.

## Mesh

Select *Extremely fine* as the element size for the *Physic-controlled mesh*.

## Simulation

**1 Model Builder → Study 1 → Step 1: Stationary → Study extensions → Optimization: X.**

**2 Compute.**

x	y
-1.00000000e+00	5.20690960e+00
-9.00000000e-01	4.40057635e+00
-8.00000000e-01	3.80676051e+00
-7.00000000e-01	3.36635356e+00
-6.00000000e-01	3.22908129e+00
-5.00000000e-01	2.63534015e+00
-4.00000000e-01	2.40507294e+00
-3.00000000e-01	2.32980312e+00
-2.00000000e-01	2.17222942e+00
-1.00000000e-01	2.01428927e+00
0.00000000e+00	1.93979953e+00
1.00000000e-01	2.11364780e+00
2.00000000e-01	2.20933975e+00
3.00000000e-01	2.32114058e+00
4.00000000e-01	2.46124910e+00
5.00000000e-01	2.68043420e+00
6.00000000e-01	3.12864151e+00
7.00000000e-01	3.41183379e+00
8.00000000e-01	3.84288950e+00
9.00000000e-01	4.36116197e+00
1.00000000e+00	5.22001330e+00

Table 3: Input data.

---

**Task 1:**

- 1 Plot the experimental data and approximated function (P) in the same plot.
  - 2 What are the approximated values of the parameters  $C_{01}$  and  $C_{10}$ .
-

---

**Task 2:**

1 Input the data from the Table 3.

2 Approximate a quadratic function

$$y = ax^2 + b \quad (1)$$

to the input data.

3 Plot the input data and approximated function in the same plot.

4 What are the approximated values of the parameters ***a*** ja ***b***.

---

# Topology Optimization

The purpose of the topology optimization is to find the number and location of voids in the structure according to the specified constraints and clear objective. Within the structural mechanics the topology optimization tries to find an answer to the question how to distribute the available material under the given load to the structure to achieve maximum stiffness.

The present example is about to find a optimal topology of the MBB (Messerschmitt-Böcklow-Blohm) beam using SIMPS (*Solid Isotropic Material with Penalization*) method. In this case it means to apply the Young modulus in the form

$$E = \mu(x)^p E_0 \quad (2)$$

where

$$0 < \mu(x) \leq 1 \quad (3)$$

Exponent  $p$  is a parameter, where  $p = 1$  corresponds to ignoring the binary behaviour of  $\mu$ , while the large values of  $p$  lead to more binary structures. The value of  $p$  is 5 in the model. Penalized Young modulus is called  $E_{SIMP}$ .

## Defining the Optimization Problem

The *Solid Mechanics* interface gives the structural properties to the beam and the *Optimization* module adds the control variables, target and constraints for the optimization problem. The stationary study is used. The beam consists of linear elastic material - structural steel. The total weight of the beam is 23550 kg, since the dimensions of the beam are 6x1x0.5 meters. The stress tensor is a function of elasticity tensor and the density. The bottom left end of the beam is fixed and the right part of the beam lies on a roller. The load is applied on the central part of the upper edge. The structure is fully defined by the elasticity tensor  $E$ , density  $\rho$  and a fixed domain. The objective functional in the model for the optimization defining the criterion of the optimality is the strain energy. The integrand for the strain energy, the strain energy density is a predefined variable, *solid.Ws* in the *Solid Mechanics* interface. The control variable is  $\rho_{design}$  (*rho\_design*), what is constrained by pointwise constraints in the range of  $10^{-4} \dots 1$  and along with integral constraint it has to be smaller than the area of the design (the whole 2D geometry of the beam) times an area fraction. The latter is a parameter *area\_frac* with the value of **0.5**.

## Modelling

- 1 Dimension: 2D.
- 2 Interfaces: Solid Mechanics (solid), Optimization (opt).
- 3 Stationary study.
- 4 Insert globalparameters according to the Table 4.
- 5 Draw a rectangle: **Width = 6**.



Name	Expression	Description
F_load	50 kN/m	Edge load
area_frac	0.5	Area fraction
p	5	Exponent in the model for Young's modulus
reg_param	100	Regularization parameter

Table 4: Global parameters.

Name	Expression	Description
E_SIMP	$2e11[\text{Pa}] * \text{mod1.rho\_design}^p$	Penalized Young's modulus
area_designdomain	$\text{mod1.intop1}(1)$	Area of design domain

Table 5: Global variables

6 Input 4 points:

- (a) (0.05,0);
- (b) (2.95,1);
- (c) (3.05,1);
- (d) (5.95,0).

7 *Model Builder* → *Model 1* → *Definitions: right button* → *Model Couplings* → *Integration* → *Source Selection: All domains*.

8 *View* → *Material Browser* → *Built-In* → *Structural steel: right button* → *Add Material to Model*.

9 *Model Builder* → *Solid Mechanics* →:

- *2D approximation* → *Plane Stress*;
- *Thickness:  $d = 0.5$* .

10 *Model Builder* → *Solid Mechanics* → *Linear Elastic Material 1* → *Linear Elastic Model* → *Young Modulus:  $E = \text{User defined, } E\_SIMP$* .

11 *Model Builder* → *Solid Mechanics: right button* → *Boundary Load* → *Boundary Selection: 5*.

12 *Boundary Load* → *Force* → *Load* →  $F_L: x = 0; y = -F\_load$ .

13 *model Builder* → *Solid Mechanics: right button* → *Fixed Constraint (Boundary Level): 2*.

14 *Model Builder* → *Solid Mechanics: Roller (Boundary Level): 7*.

15 *Model Builder* → *Optimization: right button* → *Control Variable Field*:

- *Domain Selection: All*;
- *Control variable name: rho\_design*;
- *Initial value: area\_frac*;
- *Discretization* → *Element order: Linear*.

16 *Model Builder* → *Optimization: right button* → *Pointwise Inequality Constraint (Domain Level)* →:

- *Domain Selection: All*;
- *Constraint expression: rho\_design*;
- *Bounds* → *Lower bound: 1e-4*;
- *Bounds* → *Upper bound: 1*.

17 *Model Builder* → *Optimization* → *Integral Inequality Constraint (Domain Level)* →:

- *Domain Selection: All*;
- *Constraint expression: rho\_design*;
- *Bounds* → *Upper bound: area\_designdomain\*area\_frac*.

18 *Model Builder* → *Optimization: right button* → *Integral Objective (Domain Level)*:

- *Domain Selection: All*;
- *Objective expression: solid.Ws*.

19 *Model Builder* → *Optimization: right button* → *Integral Inequality Constraint (Domain Level)*:

- *Domain Selection: All*;
- *Constraint expression:  $d(\text{rho\_design},x)^2 + d(\text{rho\_design},y)^2$*  ;
- *Bounds* → *Uppers bound: reg\_param*.

---

**Remark:** This integral inequality constraint gives a chance for the regularization to avoid the appearance of the checkerboard pattern in the optimized solution.

---

20 Insert the table of the global variables 5.

21 *Model Builder* → *Mesh 1* → *Mesh Setting* → *Element size* → *Extra fine*.

22 *Model Builder* → *Study 1* → *Step 1: Stationary* → *Study Extensions* → *Optimization: X*.

23 *Compute*.

---

**Task 3:**

- 1 Present the default plot.
  - 2 Plot the optimized structure in the reversed grayscale (“1” marks the presence and “0” the absence of the material).
-

# Bracket

The present exercise bases on the 11th episode of the 3rd season of the TV show “Rakett 69”, aired on the 23th of March 2013 in ETV. The task is to design a bracket from the given aluminum plate capable of holding the given load.

The dimensions of the aluminum plate are **5x300x500 mm**. It has to be anchored on the wall vertically with its longer edge. Other sides and edges can not have any other support. The load is applied on the opposite (outer) longer edge along the segment of 1 cm in length located symmetrically in respect of the shorter edges of the plate. The applied load is **100 kg**. The material is aluminium form the *COMSOL Multiphysics* Materials Library. Some of its parameters:

- Poisson’s ratio: 0.33;
- Young’s modulus: 70 GPa;
- yield stress: 15-20 MPa;
- ultimate tensile strength: 40-50 MPa;
- exponent  $p$  in the model of the Young’s modulus: 3.

For the parametric simulation: *Model Builder* → *Study 1* → *Job Configurations: right button* → *Parametric: right button* → *Solver*. Under *Parametric 1* add a parameter and its range of change with a step of change and specify all the combinations as a sweep type. Under *Solver 1* set *Run: Solver 1* and check *Keep all solutions*.

Solve the problem in 2D and with dense mesh.

---

## Task 4:

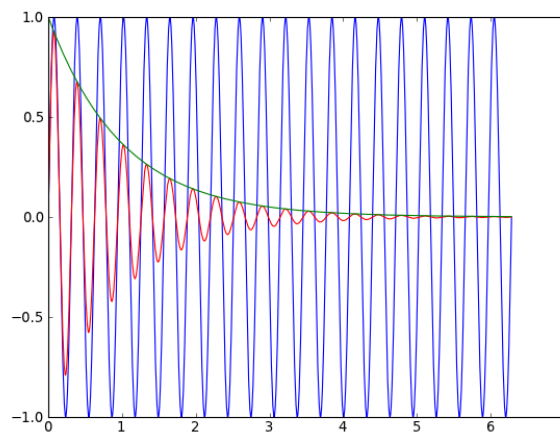
- 1 What is the minimum mass of the bracket what can stand the given load?
  - 2 What is the material area fraction of the initial plate corresponding to the minimum mass?
  - 3 Plot the material distribution with and without the deformation of the optimized structure.
  - 4 Plot the von Misses stress with and without the deformation of the optimized structure.
-

# Computational Physics II

## Practical Work

### Femhub

#### Introduction



Heiki Kasemägi

October 10, 2013

# Contents

- Introduction** **3**
- Passing the Practical Work . . . . . 3
  
- Running *FEMhub*** **4**
- Non-graphical Environment . . . . . 4
- Online Lab* in the Local Computer . . . . . 4
- Online Lab . . . . . 9
  
- Data Input/Output** **13**
- Command Line . . . . . 13
- Online Lab* . . . . . 17

# List of Figures

1	Running <i>FEMhub</i> . . . . .	5
2	Getting help in <i>FEMhub</i> . . . . .	5
3	Help in <i>FEMhub</i> . . . . .	6
4	Displaying the command code. . . . .	6
5	Starting the <i>Online Lab</i> in the local computer. . . . .	7
6	The <i>Online Lab</i> in the local computer. . . . .	7
7	Worksheet of the <i>Online Lab</i> . . . . .	8
8	Editable text mode of the worksheet. . . . .	9
9	Non-editable text mode of the worksheet. . . . .	10
10	Entering the <i>FEMhub Online Lab</i> . . . . .	10
11	Desktop of the <i>Online Lab</i> . . . . .	11
12	The desktop of the <i>Online Lab</i> with the browser, list of the published worksheets, help and settings. . . . .	11
13	Entering data from the command line. . . . .	14
14	Plot of the sine function. . . . .	14
15	Importing the module. . . . .	15
16	Calling the functions. . . . .	15
17	Start of the module code. . . . .	16
18	Module code continues. . . . .	16
19	End of the module code. . . . .	17

# Introduction

*FEMhub* (<http://femhub.org>) is a uniform *Python* (<http://www.python.org>) user interface framework for the open source programs, libraries and modules implementing the Finite Element Method. *FEMhub* is available both as standalone program as well as web service over the web browser from the computing server

*Online Numerical Methods Laboratory* (<http://lab.femhub.org>)

of the computing methods science group *hp-FEM Group* (<http://hpfem.org>) of the University of Nevada (<http://unr.edu>) in Reno (USA, Nevada).

*FEMhub* is published under the GNU General Public Licence ver. 2 (<http://www.gnu.org/licenses/old-licenses/gpl-2.0.html>),

The components of *FEMhub* are written in C/C++, Python and Fortran programming languages. *FEMhub* contains both the Python and Fortran in the form of g95 (in the case of Linux and MacOS X on x86 platform).

Some of the components of *FEMhub* originate from the software package

*Sage* (<http://www.sagemath.org/>).

## Passing the Practical Work

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading “**Task #**”. The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may rise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size, title(s), legend(s) etc. The report and accompanying files are submitted via moodle.



# Running *FEMhub*

## Non-graphical Environment

The text mode of *FEMhub* starts the command (Fig. 1):

```
$ femhub
```

The peculiarity of the *FEMhub* command line is that the commands have the brackets “()” at the end even without the argument. Help is given by the command (Fig. 2):

```
help()
```

Another way to get help is to end the command with the question mark instead of the brackets (Fig. 3). Then the help information added to the command is displayed. Two question marks instead of one display the whole code of the command if possible (Figs. 3,4).

## *Online Lab* in the Local Computer

To run the *Online Lab* of *FEMhub* first start *FEMhub* and then run the command (Fig. 5)

```
lab()
```

Then the web page opens at the address <http://localhost:8000> representing the *Online Lab*'s environment in the local computer (Fig. 6). On the first run, a password is asked for the username.

The upper row of the page displays the name of the user and a bunch of links:

**Home:** return to the home page, i.e. the list of the personal worksheets;

**Published:** worksheets what are public;

**Log:** list of the latest calculations along with the results;

**Settings:** change the settings of the user and the notebook.

The links in the next row have the following functionality:

**New Worksheet:** create a new worksheet;

**Upload:** import an existing worksheet from the local computer of the network into the *Online Lab*;

**Download All Active:** save all active worksheets on the local disk as a zip file.

```
11:39:44 marvin@maakivi:~ $ femhub
-----
| FEMhub Version 0.9.9, Release Date: 2010-05-05 |
| Type lab() for the GUI. |
-----
In [1]: █
```

Figure 1: Running *FEMhub* .

```
-----
In [1]: help()

Welcome to Python 2.6! This is the online help utility.

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

To get a list of available modules, keywords, or topics, type "modules",
"keywords", or "topics". Each module also comes with a one-line summary
of what it does; to list the modules whose summaries contain a given word
such as "spam", type "modules spam".

help> quit

You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)". Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.
In [2]: █
```

Figure 2: Getting help in *FEMhub* .

```
You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)". Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.
In [2]: quit?
Type:         Quitter
Base Class:   <class 'IPython.iplib.Qitter'>
String Form:  Type quit() to exit.
Namespace:   Python builtin
File:        /usr/local/share2/femhub-0.9.9/local/lib/python2.6/site-packages
/IPython/iplib.py
Docstring:
    Simple class to handle exit, similar to Python 2.5's.

    It handles exiting in an ipython-safe manner, which the one in Python 2.5
    doesn't do (obviously, since it doesn't know about ipython).
Call def:    quit(self)

In [3]: quit??
Type:         Quitter
Base Class:   <class 'IPython.iplib.Qitter'>
String Form:  Type quit() to exit.
Namespace:   Python builtin
```

Figure 3: Help in *FEMhub* .

```
Namespace:   Python builtin
File:        /usr/local/share2/femhub-0.9.9/local/lib/python2.6/site-packages
/IPython/iplib.py
Source:
class Quitter(object):
    """Simple class to handle exit, similar to Python 2.5's.

    It handles exiting in an ipython-safe manner, which the one in Python 2.5
    doesn't do (obviously, since it doesn't know about ipython)."""

    def __init__(self,shell,name):
        self.shell = shell
        self.name = name

    def __repr__(self):
        return 'Type %s() to exit.' % self.name
    __str__ = __repr__

    def __call__(self):
        self.shell.exit()Call def:    quit(self)

In [4]: quit()
13:17:48 marvin@maakivi:~ $
```

Figure 4: Displaying the command code.

```

12:05:09 marvin@maakivi:~ $ femhub
-----
| FEMhub Version 0.9.9, Release Date: 2010-05-05 |
| Type lab() for the GUI. |
-----

In [1]: lab()
Stub: alarm
Stub: cancel_alarm
*****
* *
* Open your web browser to http://localhost:8000 *
* *
*****

/usr/local/share2/femhub-0.9.9/local/lib/python2.6/site-packages/Twisted-9.0.0-py2.6-linux-x86_64.egg/twisted/web2/static.py:12: DeprecationWarning: the md5 module is deprecated; use hashlib instead
  import md5
2011-10-12 12:05:22+0300 [-] Log opened.
2011-10-12 12:05:22+0300 [-] twisted 9.0.0 (/usr/local/share2/femhub-0.9.9/local/bin/python 2.6.4) starting up.
2011-10-12 12:05:22+0300 [-] reactor class: twisted.internet.selectreactor.SelectReactor.
2011-10-12 12:05:22+0300 [-] twisted.web2.channel.http.HTTPFactory starting on 8000

```

Figure 5: Starting the *Online Lab* in the local computer.

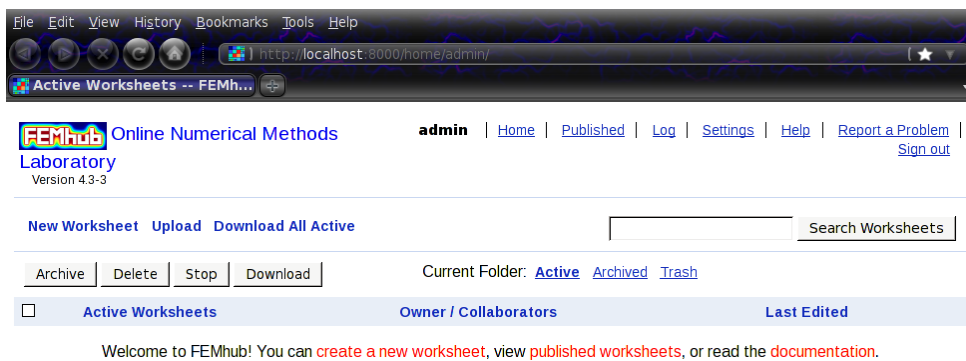


Figure 6: The *Online Lab* in the local computer.

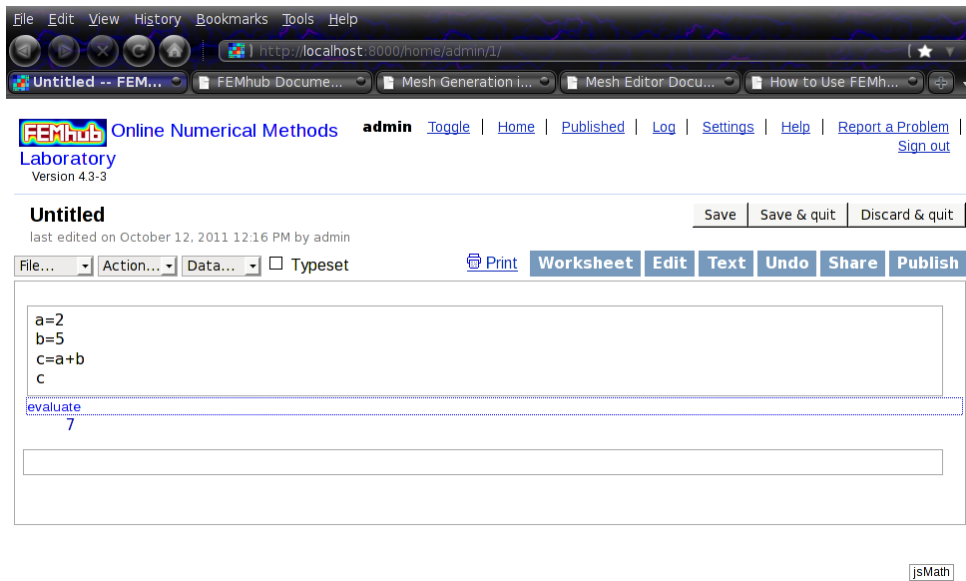


Figure 7: Worksheet of the *Online Lab*.

The buttons in the next row make possible the operations with the worksheets:

**Archive:** archiving the selected worksheets, which does not appear in the list of the worksheets any more;

**Delete:** deleting the selected worksheets, what means moving the worksheets into the “Trash” directory;

**Stop:** stopping the selected worksheets;

**Download:** downloading the selected worksheets on the local disk in zip format.

A fresh worksheet (Fig. 7) has a button “Toggle” in the upper menu what can switch on/off the menubar below. The operations what are available via this menu include saving the worksheet (“Save”), saving the worksheet with quitting the environment (“Save & quit”), exiting the environment without saving the worksheet (“Discard & quit”), importing the worksheet from a file (“File... →Load worksheet from a file...”), creating a new worksheet (“File... →New worksheet”), saving the worksheet into a file (“File... →Save worksheet to a file...”), printing the worksheet (“File... →Print”), renaming, copying and deleting the worksheet (“File... →Rename worksheet”, “File... →Copy worksheet”, “File... →Delete worksheet”).

Operations through the “Action” menu:

**Interrupt:** stop the running calculations if needed;

**Restart worksheet:** rerun the calculations;

**Save and quit worksheet:** save and exit the worksheet;

**Evaluate All:** calculate all input cells in the worksheet;

**Hide All Output:** hide all results;

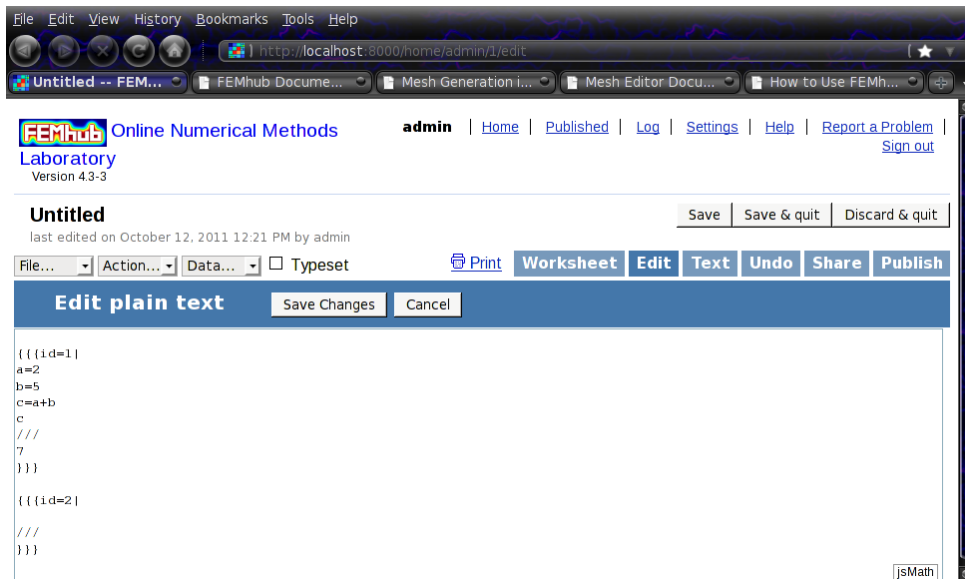


Figure 8: Editable text mode of the worksheet.

**Show All Output:** show all results;

**Delete All Output:** delete all results;

**One Cell Mode:** worksheet with a single input cell;

**Multi Cell Mode:** worksheet with multiple input cells.

**Worksheet:** return to the interactive mode of the worksheet (Fig. 7);

**Edit:** editable text mode of the worksheet (Fig. 8);

**Text:** switches to the non-editable text mode (Fig. 9);

**Undo:** displays the changes in the worksheet;

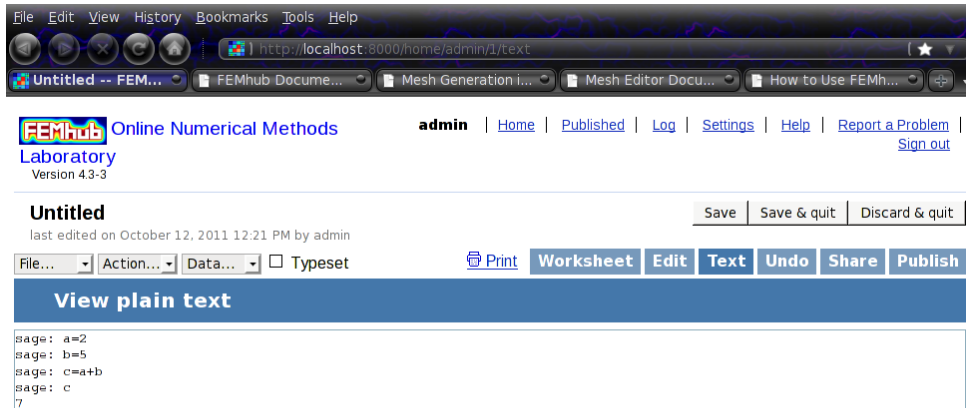
**Share:** shares the worksheet for editing to other users;

**Publish:** makes the worksheet public.

## Online Lab

Open network address <http://lab.femhub.org> to use the *Online Lab* of *FEMhub* and create an user account (Fig. 10). After the entering a desktop-like environment opens (Figs. 11, 12):

**Browser:** managing of the worksheets: creating a new one, saving or relocating between the directories, editing, renaming, deleting. Mouse can be used to move the worksheets between the directories and directories between directories. If any of the operations is not allowed, the appropriate message will be displayed. The *Browser* has the following buttons:



jsMath

Figure 9: Non-editable text mode of the worksheet.

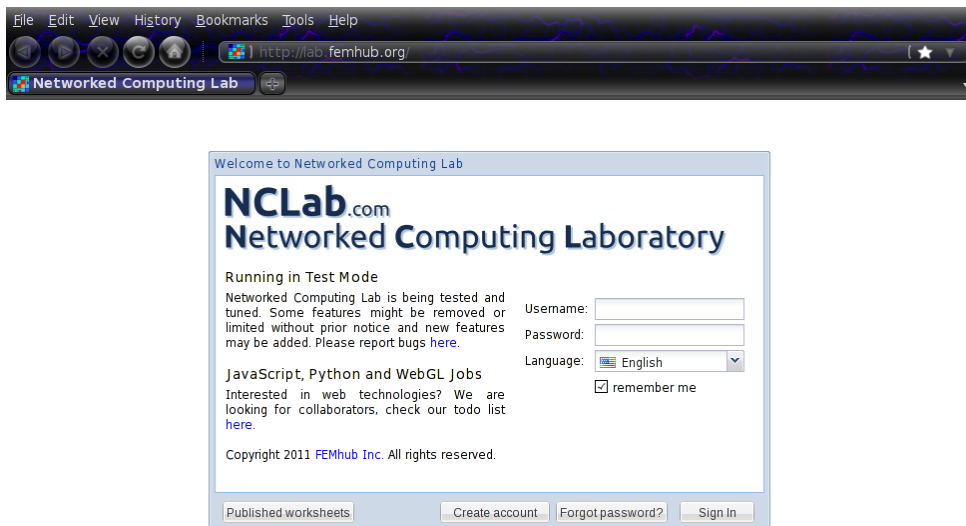


Figure 10: Entering the *FEMhub Online Lab*.

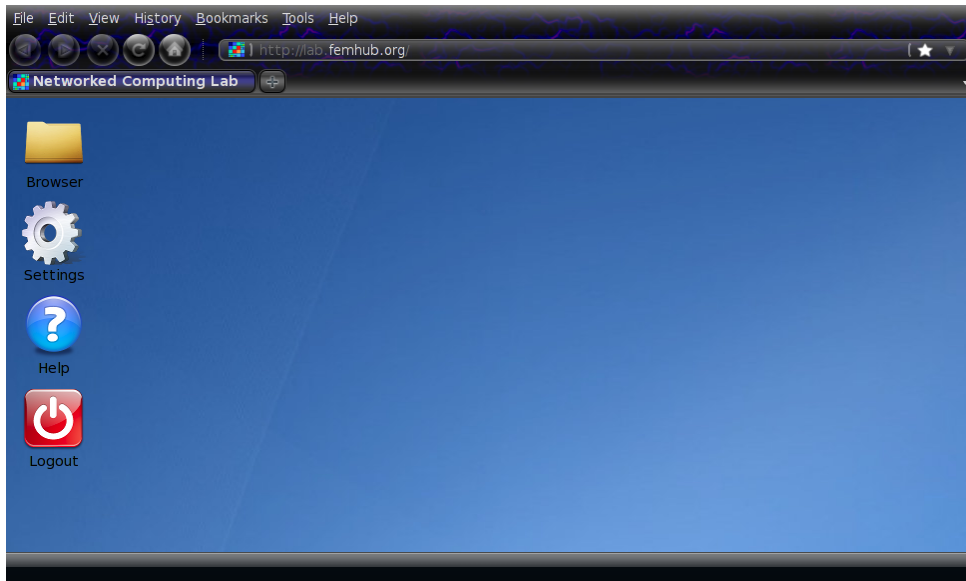


Figure 11: Desktop of the *Online Lab*.

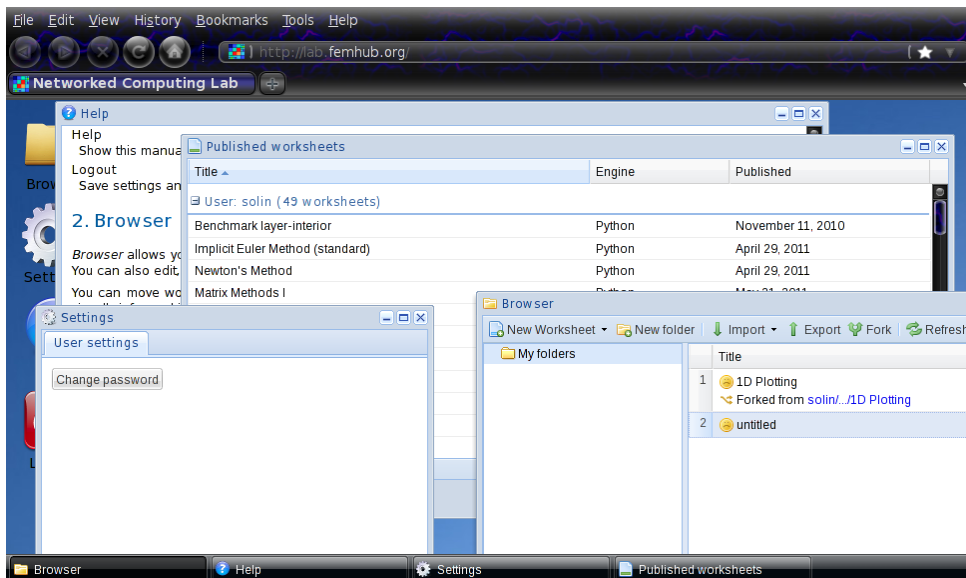


Figure 12: The desktop of the *Online Lab* with the browser, list of the published worksheets, help and settings.



**New Worksheet:** create a new worksheet;

**New Folder:** create a new directory;

**Import Worksheet:** import a worksheet from the RST and SAGE formats;

**Fork Worksheet:** displays a list of the public worksheets. The “View” button displays the selected worksheet, ”Fork” button saves a local copy of the public worksheet.

**Refresh:** refreshing the browser;

**Rename:** renaming selected directories or worksheets;

**Delete:** deleting selected directories or worksheets.

**Settings:** user settings (changing the password);

**Help:** help;

**Logout:** saving the settings and ending the current session.

# Data Input/Output

## Command Line

Entering the data from the command line is similar to other programs having a command line (Fig. 13). The result of the commands in Fig. 14 is a plot of the sine function.

Another way is to create a py-file with the modules and functions within the modules. The py-file can be imported to *FEMhub* like any other module and used in the similar way (Fig. 15). After starting *FEMhub* one has to ensure that the current location is correct. This is done by the command

```
pwd
```

what displays the working directory. To return the home directory:

```
cd
```

Since the file *sine.py* is in the home directory, import it:

```
import sine
```

One must be noticed that *sine* is a name of the module. The function inside the module is called in the following way (Fig. 16)

```
module.function[.function[...]]
```

To import something, that has to exist. The commented code of the *sine* module and *sine.sine* function is in the Figs. 17, 18, and 19. It has to be noticed that the text of the code must be hierarchical, i.e. indentation determines the relations between the structural elements.

In order to calculate in the *FEMhub* environment, the proper modules and functions need to be imported. For example, the *numpy* contains the arrays of the arbitrary homogeneous objects, fast mathematical operations over those arrays, linear algebra functions, fast Fourier transform, and a random number generator.

The module can be imported in several ways. In the Fig. 17 the functions are called with a command

```
module.function()
```

like

```
numpy.pi  
numpy.sin(t)  
pylab.clf()
```

```
13:11:54 marvin@tommy:~ $ femhub
-----
| FEMhub Version 0.9.9, Release Date: 2010-05-05 |
| Type lab() for the GUI. |
-----
In [1]: import pylab, numpy
In [2]: tmin=0
In [3]: tmax=2*numpy.pi
In [4]: timestep=0.001
In [5]: t=numpy.arange(tmin,tmax,tstep)
In [6]: f=20
In [7]: sine=numpy.sin(f*t)
In [8]: pylab.plot(t,sine,"b-")
Out[8]: [<matplotlib.lines.Line2D object at 0x3a348d0>]
In [9]: cd /home/marvin
/home/marvin
In [10]: pylab.savefig("sine.png")
In [11]: _
```

Figure 13: Entering data from the command line.

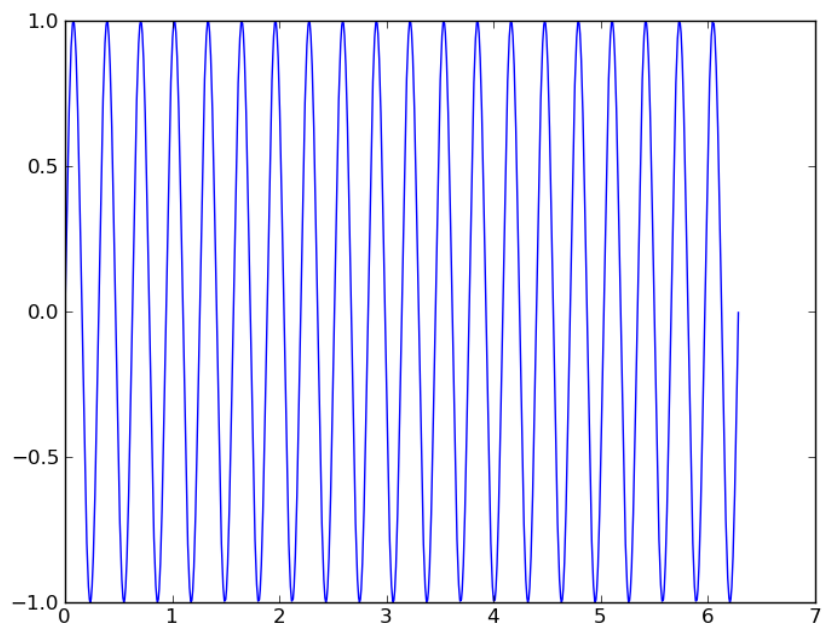


Figure 14: Plot of the sine function.

```
13:22:24 marvin@tommy:~ $ clear
13:22:26 marvin@tommy:~ $ femhub
-----
| FEMhub Version 0.9.9, Release Date: 2010-05-05
| Type lab() for the GUI.
-----
In [1]: pwd
Out[1]: '/usr/local/share2/femhub-0.9.9/local/bin'
In [2]: cd /home/marvin
/home/marvin
In [3]: import sine
In [4]: sine?
Type:          module
Base Class:    <type 'module'>
String Form:   <module 'sine' from 'sine.py'>
Namespace:    Interactive
File:         /home/marvin/sine.py
Docstring:
    Module to create the graph of the sine function

    Functions: sine
In [5]: sine.sine?
```

Figure 15: Importing the module.

```
String Form:   <module 'sine' from 'sine.py'>
Namespace:    Interactive
File:         /home/marvin/sine.py
Docstring:
    Module to create the graph of the sine function

    Functions: sine
In [5]: sine.sine?
Type:          function
Base Class:    <type 'function'>
String Form:   <function sine at 0x1c58050>
Namespace:    Interactive
File:         /home/marvin/sine.py
Definition:    sine.sine(f=20, fail='sine.png')
Docstring:
    Function to create the graph of the sine function
    Arguments: f - oscillation frequency
               fail - file to store the plot
In [6]: sine.sine??
In [7]: sine.sine(f=20, fail="/tmp/sine.png")
sine ready...
In [8]:
```

Figure 16: Calling the functions.

```
"""
Module to create the graph of the sine function

Functions: sine
"""

def sine(f=20, fail="sine.png"):

    """
    Function to create the graph of the sine function
    Arguments: f - oscillation frequency
               fail - file to store the plot
    """

    import pylab, numpy

    def calc():
#start of the time line
        tmin=0
#end of the time line
        tmax=2*numpy.pi
#time step
        tstep=0.001
```

Figure 17: Start of the module code.

```
#time step
        tstep=0.001
#create the time line from tmin upto tmax with the step of tstep
        t=numpy.arange(tmin, tmax, tstep)
#create the sine
        sine=numpy.sin(f*t)
#clear the graph
        pylab.clf()
#creating the graph, b- makes a solid blue line
        pylab.plot(t, sine, "b-", label="sine")
#add the title
        pylab.title("Sine graph")
#add the legend
        pylab.legend()
#add x-axis title
        pylab.xlabel("Time /s")
#add y-axis title
        pylab.ylabel("Amplitude")
#add the grid
        pylab.grid()
#set x-axis limits
        pylab.xlim(numpy.min(t), numpy.max(t))
#set y-axis limits
```

Figure 18: Module code continues.

```
#creating the graph, b- makes a solid blue line
pylab.plot(t,sine,"b-",label="sine")
#add the title
pylab.title("Sine graph")
#add the legend
pylab.legend()
#add x-axis title
pylab.xlabel("Time /s")
#add y-axis title
pylab.ylabel("Amplitude")
#add the grid
pylab.grid()
#set x-axis limits
pylab.xlim(numpy.min(t),numpy.max(t))
#set y-axis limits
pylab.ylim(numpy.min(sine),numpy.max(sine))
#save the graph into the file
pylab.savefig("fail")
#announce the completion
print "sine ready..."
#call the function
calc()
```

Figure 19: End of the module code.

The module can be imported as a variable, like

```
import numpy as np
import pylab as p
```

and called as

```
np.exp(-t)
p.savefig("test.png")
```

The third way is to call only necessary functions:

```
from numpy import sin, cos, exp, arange, pi
from pylab import plot, savefig
```

These functions are called directly by their names:

```
t=arange(0,2*pi,0.001)
y=sin(t)
plot(t,y,"b-")
```

User defined variables are listed by the command

```
who
```

## Online Lab

To generate the sine graph in the **Online Lab** click the *Import* button in its browser and copy the following code into the dialogue window.

====

Introduction to the **Online Lab**

====

It is possible to add the formatted comments in the worksheet of the **Online Lab** besides the regular calculations. The comments use **ReStructuredText** syntax. The information about the syntax is available in the web address

<http://docutils.sourceforge.net/docs/user/rst/quickstart.html>

[http://docutils.sourceforge.net/docs/ref/rst/ \ restructuredtext.html](http://docutils.sourceforge.net/docs/ref/rst/restructuredtext.html)

**\*\*NB:** Do not click these links without saving the worksheet, because then opened webpage closes the Online Lab along with the worksheet and the unsaved changes will be lost.\*\*

The code can be entered in three ways:

- (1) select **Fork** in the browser and copy the public worksheet into your own browser;
- (2) select **Import** in the browser to copy the code into the dialogue window;
- (3) enter the code and the comments into the text and code cells of the new worksheet.

But now the real example...

Sine

=====

The graph of the sine is one of the simplest task. The general shape of the sine function is

$$y(t) = A \cdot \sin(w \cdot t + f),$$

where  $A$  – maximum amplitude,  $w$  – oscillation frequency,  $f$  – phase shift.

```
{{{
```

```
#lets start with the definition of the function what generates the sine graph
```

```
def sine(tmin=0,tmax=2,tstep=0.001,w=20,f=0):
```

```

#import modules pylabi and numpy
import pylab ,numpy
#define minimum time limit
tmininp=tmin*numpy.pi
#define maximum time limit
tmaxinp=tmax*numpy.pi
#create the time axis
t=numpy.arange(tmininp ,tmaxinp ,tstep)
#calculate the sine values
sine=numpy.sin(w*t+f)
#clear the plot
pylab.clf()
#plot the graph
pylab.plot(t ,sine ,"b-",label="sine ")
#display the legend
pylab.legend()
#show the graph
lab.show()
#notifiy the the program terminates
print "sine ready"
}}

```

If the code is entered , click on the *\*evaluate\** button under the input cell. Then the content of the cell is calculated and notified about the possible errors of the result. If debugging or changing the code, this button can be used as many times as needed. One has to keep in mind that evaluation results from a single cell does not propagate automatically to the following cell, which has to be recalculated separately , except after clicking on the *\*Evaluate All\** button.

If the code is debugged, it can be run. If the code has no *\*def\** records , the results are available immediately after the successfull calculation. This case requires to call the *\*sine\** function. The simplest way is to call it with default input parameters:

```

{{{
sine()
///
sine ready
///
e2c87680df3d491691654ddc55637fe1
}}}

```

The parameters can be passed to the function singly or multiple. If there is more than one input parameter , then the value has to be input as

```

*parameter_name=parameter_value*:
{{{

```



```

sine(w=10,tmin=1)
///
sine ready
///
2fcaafc30c0745d09e7b0423768a40bb
}}}

```

This code can be saved into the local computer, if select worksheet in the browser and click the \*Export\* button. Copy the worksheet code from the opened window. The plots can be saved by the usual browser methods.

**\*\*NB:** The same warning applies to the right mouse click on the "View Image" as for clicking the web links:\*\*

**\*\*the worksheet will be closed without saving the changes and the chosen picture will be displayed...\*\***  
 So, it's time to add here your solution...;)

In the worksheet using this structure and syntax, everything outside the triple brackets “{{{ }}}” is the ordinary text, which is formattable as a *ReStructuredText* at the address (<http://docutils.sourceforge.net/docs/user/rst/quickstart.html>).

The executable code is within brackets. Evering after the triple slash “///” until the next triple slash or terminating brackets “}}}" is an output cell. There is no need to enter the output code into the worksheet since the code probably needs to be rerunned. Besides, the output may contain identifier of the images and other object, which are updated on every run.

---

### Task 1:

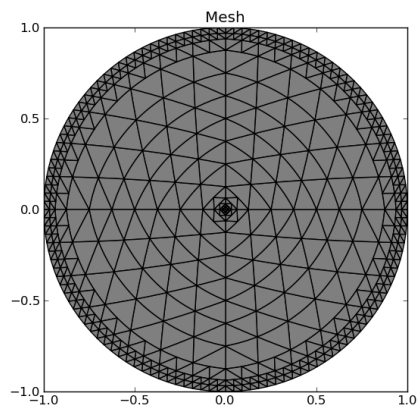
- 1 Generate the plot in Fig. 14 based on the code in Fig. 13.
  - 2 What are the results of the executing the code in Fig. 17...19?
  - 3 Add a new function at the end of the sine worksheet what complements the code of the sine function to execute the following tasks:
    - generates the sine wave, the damped sine wave and wrapping line in the same plot;
    - inputs values for the amplitude, frequency and phase shift for the sine wave;
    - inputs the damping factor for the damped sine wave;
    - adds labels for the plot and the axes;
    - adds a legend;
    - determines atomatically the minimum and maximum limits of the axes;
    - plots the result of the new code.
-

# Computational Physics II

## Practical Work

### FEMhub

### Meshing



Heiki Kasemägi

October 10, 2013

# Contents

- Meshing** **3**
- Introduction . . . . . 3
- Passing the Practical Work . . . . . 3
- References . . . . . 3
- L-shaped Object . . . . . 4
- Importing the mesh from the Hermes File, . . . . . 6
- Importing the Mesh from the External Formats . . . . . 8
- Refining the Mesh . . . . . 9

# List of Figures

1	L-shaped example object. . . . .	4
2	Creating the mesh of the L-shaped object. . . . .	5
3	Plotting the mesh of the L-shaped object. . . . .	5
4	The code to refine and plot the mesh. . . . .	9
5	Uniformly refined mesh. . . . .	10
6	Element refinement. . . . .	10

# Meshing

## Introduction

The Finite Element simulation starts with dividing the domain into the finite elements. *FEMhub* uses triangular and quadrilateral elements from *Hermes*. These element types can be mixed within the same mesh. The arcs can be placed on the sides of the elements, if possible. Triangular elements are suitable to solve isotropic problems, quadrilateral ones are good for anisotropic cases (like surface layers etc.). *Hermes* is able to import a complex mesh generated by the external specialised software in the proper file format. If the complexity is not high, the mesh can be created manually with *Hermes* tools. In *FEMhub* and *Hermes*, it only needs to mesh the domain with a very coarse mesh and let the adaptive function do the rest of the work.

## Passing the Practical Work

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading “**Task #**”. The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may rise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size, title(s), legend(s) etc. The report and accompanying files are submitted via moodle.

## References

The present guide and the practical work base on the Hermes documentation  
<http://hpfem.org/hermes-doc/hermes/html/index.html>.

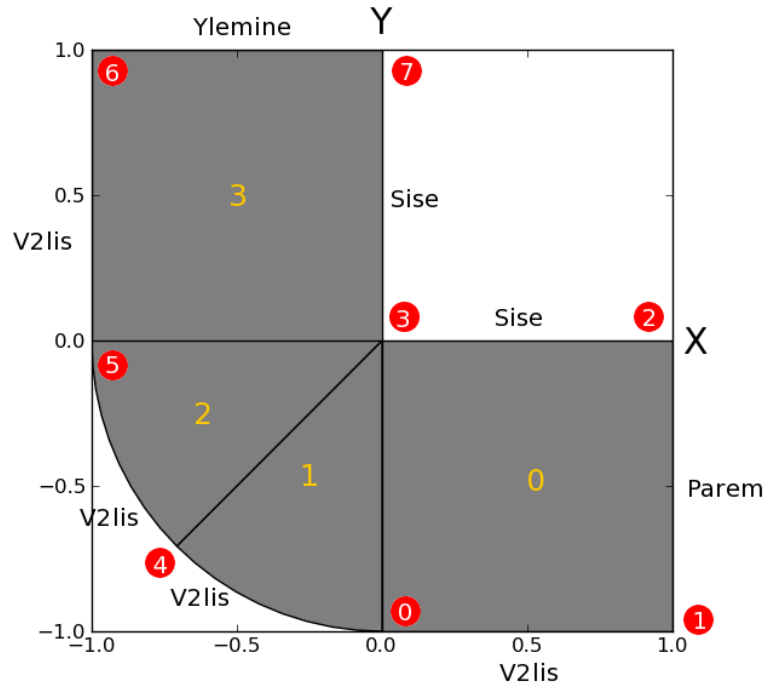


Figure 1: L-shaped example object.

## L-shaped Object

The example object has a L-shape (Fig. 1). It has 4 elements: 2 quadrilateral and 2 triangular elements. Triangular element has an arc. The nodes are marked by the numbers on the red circle, elements by the yellow numbers. “X” and “Y” mark the positive direction of the coordinate axes. In principle, every element may consist of different materials. To create the mesh for this object, open the *Online Lab* in the local computer and create a new worksheet. The first step is to import the mesh-related functions from the *hermes2d* module (Fig. 2). The value of the *mesh* variable is a new object that contains the mesh. The *mesh.create()* method defines the nodes, elements, edges and arcs, if needed. The format is a list of the lists with nodes, edges, etc. as the components. The components of the list are:

**nodes:** the definition of the each node consists of a pair of the x and y coordinate. The position of the each node marks the index of the node. The indexation starts from zero;

**elements:** the list of the elements contains all the elements in the preliminary mesh. The element is defined as a sequence of the element indexes, what forms the element, and the indexes are listed counterclockwise. Besides, an additional string or positive integer is added after the element index to describe the material. Materials indication allows to use different weak forms in the subdomains. This is useful if the domain consists of different materials, but in this way it is possible to define completely different physical processes in the subdomains. The element and edge marks are available via weak form. Integer marks does not have to be in the brackets;

**edges:** this is the last mandatory element to mark all edges. An edge is marked by the triple of numbers: two nodes and an edge mark in the form of a string or an integer. The mark can not be zero or negative integer, and the boundary conditions are applied to the edges

```

File Edit View History Bookmarks Tools Help
localhost:8000/home/admin/7/
Untitled -- FEMhub

# import the necessary modules
from hermes2d import Mesh,MeshView

mesh=Mesh()
mesh.create([
# defining the nodes
  [0,-1], # node 0
  [1,-1], # node 1
  [1,0], # node 2
  [0,0], #node 3
  [-0.707106781,-0.707106781], # node 4 at the edge
  [-1,0], # node 5
  [-1,1], # node 6
  [0,1] # node 7
],[
# defining the elements
  [0,1,2,3,0], # element 0
  [0,3,4,0], # element 1
  [3,5,4,0], # element 2
  [3,7,6,5,0] # element 3
],[
# defining the edges
  [0,1,1], # edge 0
  [1,2,2] # edge 1
]

```

Figure 2: Creating the mesh of the L-shaped object.

```

File Edit View History Bookmarks Tools Help
localhost:8000/home/admin/7/
Untitled -- FEMhub

[3,5,4,0], # element 2
[3,7,6,5,0] # element 3
],[
# defining the edges
  [0,1,1], # edge 0
  [1,2,2], # edge 1
  [2,3,3], # edge 2
  [4,0,1], # edge 3
  [5,4,1], # edge 4
  [6,5,1], # edge 5
  [6,7,4], # edge 6
  [3,7,3] # edge 7
],[
# defining the arcs
  [4,0,45], # arc 0
  [5,4,45] # arc 1
])
mesh.plot(filename="mesh.png")

```

Mesh




Figure 3: Plotting the mesh of the L-shaped object.

in the form of the positive integer. The negative integers can be used for internal edges to form the arcs of them, but it is suggested not to exaggerate with the arcs, since they increase the cost of the numerical integration and thus the calculation time.

**arcs:** these are the optional elements and if they are not present, just add the empty brackets “[]” to mark an empty list. Each arc is defined by the single *Non-Uniform Rational B-Spline*’iga (NURBS ([http://en.wikipedia.org/wiki/Non-uniform\\_rational\\_B-spline](http://en.wikipedia.org/wiki/Non-uniform_rational_B-spline))), consisting of an angle, the control points and a knot vector. The most common arc is a quadrant of the circle and it is defined by the central angle and the control points at the ends of the arc (Fig. 3).

If the mesh is defined, it has to be displayed to ensure if it is correct (Fig. 3).

---

**Task 1:** Plot the mesh defined in Figures 2 and 3

---

## Importing the mesh from the Hermes File,

**Hermes** can handle different file formats. Here, the attention is on its own file format.

The comments start with a hash “#”. The *Hermes* mesh file format consists of the definitions of the variables. The variables can be a real numbers, lists of the real numbers or the lists of the lists.

```
# this is a comment
number = 45.0 + sin(pi) # ordinary number
objekt = {0,1,2,3,number} # list
pairs = {{0,5},{1,number},{2,objekt}} # list of the lists
```

The following mesh file describes the mesh in the Fig. 1:

```
zero = 0 # variable definition
step = 1
arc = sqrt(2)/2

# node definitions
vertices =
{
  {0,-1}, # node 0
  {step,-step}, # node 1
  {step,zero}, # node 2
  {zero,zero}, # node 3
  {-arc,-arc}, # node 4 at the edge
  {-step,zero}, # node 5
  {-step,step}, # node 6
  {zero,step} # node 7
}
# element definition
elements =
```



```

{
  {0,1,2,3,0}, # element 0
  {0,3,4,0}, # element 1
  {3,5,4,0}, # element 2
  {3,7,6,5,0} # element 3
}
# edge definition
boundaries =
{
  {0,1,1}, # edge 0
  {1,2,2}, # edge 1
  {2,3,3}, # edge 2
  {4,0,1}, # edge 3
  {5,4,1}, # edge 4
  {6,5,1}, # edge 5
  {6,7,4}, # edge 6
  {3,7,3} # edge 7
}
# arc definition
curves =
{
  {4,0,45}, # arc 0
  {5,4,45} # arc 1
}
# EOF

```

Square brackets are also accepted:

```

zero = 0 # variable definition
step = 1
arc = sqrt(2)/2
outer = 1
right = 2
inner = 3
upper = 4

# node definition
vertices =
[
  [0,-1], # node 0
  [step,-step], # node 1
  [step,zero], # node 2
  [zero,zero], # node 3
  [-kaar,-kaar], # node 4 at the edge
  [-step,zero], # node 5
  [-step,step], # node 6
  [zero,step] # node 7
]
# element definition
elements =

```

```

[
  [0,1,2,3,0], # element 0
  [0,3,4,1], # element 1
  [3,5,4,1], # element 2
  [3,7,6,5,0] # element 3
]
# edge definition
boundaries =
[
  [0,1,outer], # edge 0
  [1,2,right], # edge 1
  [2,3,inner], # edge 2
  [4,0,outer], # edge 3
  [5,4,outer], # edge 4
  [6,5,outer], # edge 5
  [6,7,upper], # edge 6
  [3,7,inner] # edge 7
]
# arc definition
curves =
[
  [4,0,45], # arc 0
  [5,4,45] # arc 1
]
# EOF

```

If the variables are defined with a line break, like

```

vertices =
{

```

than there should be no whitespaces after the “=” at the end of the row.

If the **Online Lab** is running in the local computer, copy the mesh file into the directory:

```
$HOME/.femhub/sage_notebook.sagenb/home/admin/2/data
```

where *\$HOME* is the user home directory in the local computer, *admin* should be replaced by the user name in the **Online Lab**, and 2 by the number of the existing worksheet what imports the mesh. The code to import the mesh:

```

test=Mesh()
test.load_hermes("data/sample.h2d-kant.mesh")
test.plot(filename="test.png")

```

## Importing the Mesh from the External Formats

*Hermes* can import ExodusII-format (<http://sourceforge.net/projects/exodusii/>). Also, the *Hermes* XML format is accepted.

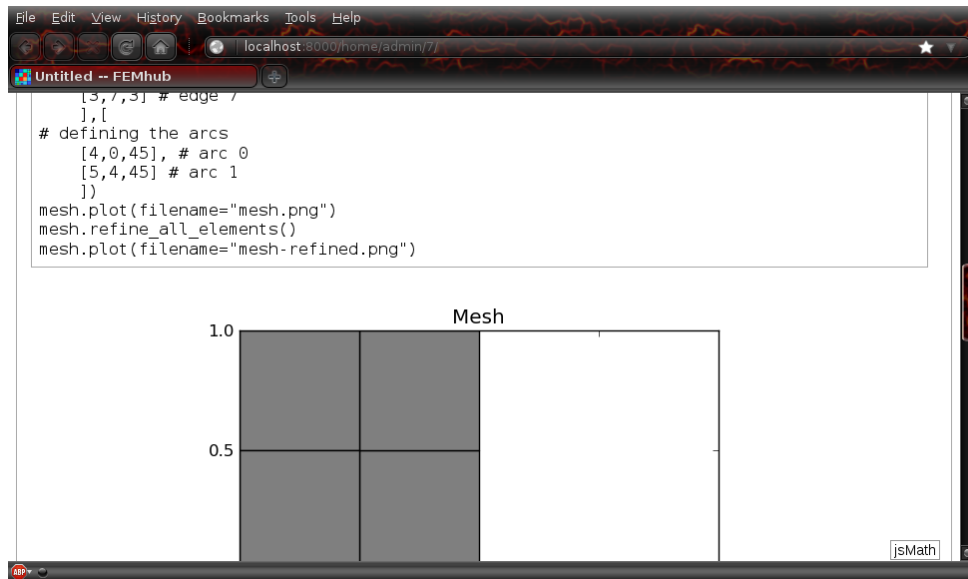


Figure 4: The code to refine and plot the mesh.

## Refining the Mesh

In the examples above the mesh is quite coarse. The calculation with this kind of mesh will probably not converge to the solution. The coarse mesh can be refined by the function `refine_all_elements()` from the `Mesh` class (Fig. 4), resulting in the finer mesh (Fig. 5). The function `refine_all_elements()` refines the mesh uniformly. The function can be applied several times in the row. Usually, the result of the refinement depends on the order of the different refinement operations.

The elements can be refined one-by-one. The appropriate function is

```
refine_element(index, mode)
```

where the *index* is the index of the element under the refinement and the value of the *mode* is either 0 (uniform refinement in all directions), 1 (horizontal) or 2 (vertical refinement) (Fig. 6). During every refinement step, one has to keep in mind that every step changes the number of the elements and therefore the indexation of the elements changes.

The function `refine_towards_vertex(index, depth)` refines the mesh towards the node *index* *depth* times.

The function `refine_towards_boundary(mark, depth)` refines the elements adjacent to the edge *mark* *depth* times.

The function `convert_quads_to_triangles()` converts the quadrilateral elements into the triangle elements.

The function `convert_triangles_to_quads()` converts the triangle elements into the quadrilateral elements.

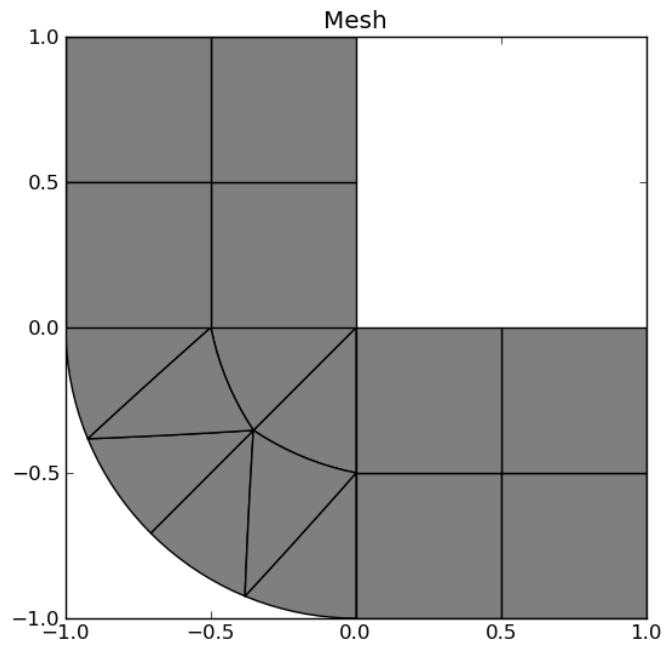


Figure 5: Uniformly refined mesh.

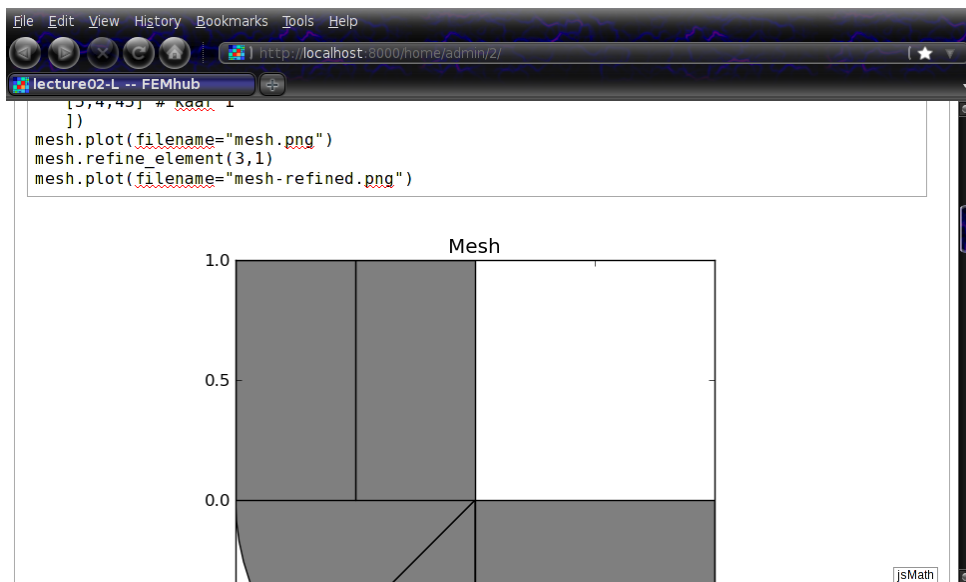


Figure 6: Element refinement.

---

**Task 2:**

- 1 Refine the initial mesh of the L-shaped objects 3 times uniformly.
  - 2 Plot the refined mesh.
  - 3 What are the changes in the element count.
- 

---

**Task 3:**

- 1 Refine all of the elements of the initial mesh of the L-shaped object once uniformly.
  - 2 Then refine the right bottom quadrilateral element once uniformly.
  - 3 Then refine the upper left element once horizontally.
  - 4 Plot the final result and all intermediate results.
- 

---

**Task 4:**

- 1 Refine the initial mesh of the L-shaped object towards the coordinate starting point three times.
  - 2 Refine the initial mesh of the L-shaped object towards the edge 1 twice.
  - 3 Convert the elements of the initial mesh of the L-shaped object at first from quadrilateral form into the triangular shape and then the resulting triangular elements back into the quadrilateral shape.
  - 4 Plot all results.
- 

---

**Task 5:**

- 1 Create a L-shaped object with rounded ends and rounded inner and outer corners. The radius of the curvature must be no more than an half of the width of the L-s appropriate branch.
  - 2 Mesh the object twice uniformly and additionally refine the curvatures uniformly once.
  - 3 Plot the result and the entire code to accomplish the task.
-

---

**Task 6:**

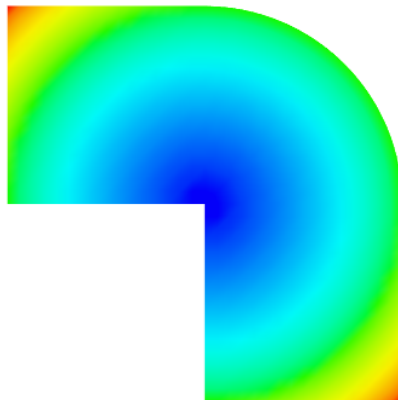
- 1 Create an unit circle.
  - 2 Refine the mesh uniformly three times and then twice in the center and twice at the edge.
  - 3 Plot the result and the entire code to accomplish the task.
-

# Computational Physics II

## Practical Work

FEMhub

Hermes2D



Heiki Kasemägi

October 3, 2013

# Contents

- Hermes2D** **3**
- Introduction . . . . . 3
- Passing the Practical Work . . . . . 3
- References . . . . . 3
- Hermes2D* Properties . . . . . 4
- Forming the Finite Element Space . . . . . 4
- Example Problem: Poisson Equation . . . . . 5



# List of Figures

1	Initial mesh of the example problem. . . . .	6
2	Refined mesh of the example problem. . . . .	7
3	The solution of the example problem. . . . .	9
4	The ranks of the mesh elements of the example problem. . . . .	9

# Hermes2D

## Introduction

*Hermes2D* is a free C++/Python library under the GPLv2, what constains tools to develop solvers for the adaptive hp-FEM and hp-GD to solve partial differential equations (*PDEs*) and multiphysical PDE systems.

The standard way to use *Hermes2D* is to write a short C++ program, but a graphical user interface called Agros2D (<http://hpfem.org/agros2d/>) is also available. In addition, *Hermes2D* can be used in the *Online Lab* of the *FEMhub*.

## Passing the Practical Work

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading “**Task #**”. The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may rise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size, title(s), legend(s) etc. The report and accompanying files are submitted via moodle.

## References

The present guide and the practical works base on the Hermes documentation <http://hpfem.org/hermes-doc/hermes/html/index.html>.

## ***Hermes2D* Properties**

**Mature hp-adaptive algorithms.** *Hermes* pays a great attention to the error control and automatic adaptivity. From the real-life experiences it is known that combining the automatic adaptivity with the standard low-level approximations like a linear and quadratic elements needs too large effort - the error decreases somehow during the few initial adaptation steps, but then decreasing slows down and even adding new unknowns and the extension of the computational time do not have a positive effect. This is common to the low-level methods. Exponentially converging adaptive hp-FEM and hp-GD are free of this problem, because the decrease of the error is stable and fast through the whole adaptation until the desired accuracy is achieved.

**Wide range of the exploitation.** *Hermes* is independent of the PDEs. The standard FEM libraries are created to solve some specific class of the problems like elliptical equations, flow dynamics, electromagnetism etc. *Hermes* on the contrary does not contain any specific technical trick or algorithm, what would restrict its use to solve any specific PDE problem. The automatic adaptation is driven by the universal error estimation based on the facts, and what is valid in a unique way for an arbitrary PDE. This does not mean that the algorithm is efficient for all PDEs in the same way - solving some of the equations is just a little more complex than others. Besides handling an arbitrary PDE or a multiphysical system of the PDEs, it is possible to add the problem- and equation-specific extensions by the user into *Hermes*.

**Nodes on the arbitrary level.** *Hermes* adopts an unique original method to handle non-regular meshes containing the nodes at the arbitrary levels. This means that the very small elements can be adjacent to the very large elements. While the elements are refined, the neighbouring elements are never refined by default like it happens then using the conventional adaptive algorithms. It is well known that the approximations with the nodes on the same level are more effective than regular meshes. The technique of the nodes on the different level makes these approximation methods ideal.

**Multimesh hp-FEM.** In the case of the multiphysical problems, the different physical quantities or the components of the solutions can be approximated within the individual meshes by combining the elements of the different rank. Splitting the operators and transferring the data does not result in any additional error.

**Dynamical meshes of the time-dependent problems.** In the case of the time-dependent problems, the physical quantities of the components of the solutions can be approximated within the individual meshes, what are independent of each other. The transferring of the solution data between the different meshes and time-moments adds no additional error.

## **Forming the Finite Element Space**

*Hermes* follows tightly the mathematical conception of the FEM, which requires the creation of the finite element space after the generating the mesh. The following predefined spaces exist at the moment:

- *H1Space* - this is the most common set of the partially continuous polynomial functions belonging to the  $H^1(\Omega) = \nu \in [L^2(\Omega); \nabla u \in [L^2(\Omega)]^2$  space;

- *HcurlSpace* - the space of the vector functions what are non-continuous along the mesh boundaries and with continuous tangential components at the boundaries:  
 $H(curl, \Omega) = E \in [L^2(\Omega)^2; \nabla \times E \in L^2(\Omega)];$
- *HdivSpace* - the space of the vector functions what are non-continuous along the mesh boundaries and with continuous normal components at the boundaries:  
 $H(div, \Omega) = \nu \in [L^2(\Omega)^2; \nabla \cdot \nu \in L^2(\Omega)];$
- *L2Space* - functions, what are non-continuous along the boundaries and belong to the  $L^2(\Omega)$  space.

All this spaces contain higher-rank elements and the meshes with the nodes on the arbitrary level. Higher-rank elements mean that the spaces contain quadratic, cubic and other boundary functions, what are able to generate higher-degree polynomials along the mesh boundaries and the “bubble”-functions, what complete the higher-degree approximations inside the elements. The boundary functions are related to the boundaries of the mesh and the “bubble”-functions to the inner parts of the elements.

There are several ways to define higher rank base functions. The certain set of the polynomials is called a *shaperset*. Using a good shaperset is quite a decisive for the performance of the hp-FEM. All shape functions could not be optimal for all possible operators. Therefore, **Hermes** contains several shapesets and the user can choose a suitable one to build a finite element space. The experience indicates that the best spaces for most of the calculations are *H1Shaperset*, *HcurlShaperset*, *HdivShaperset* and *L2Shaperset*. Every shaperset can be used in more than one space.

## Example Problem: Poisson Equation

The example problem to solve is the Poisson equation

$$-\nabla u = CONST_F \quad (1)$$

in domain  $\Omega$  with homogeneous (zero) Dirichlet boundary conditions

$$u = 0 \quad in \quad \delta\Omega \quad (2)$$

Here the  $CONST_F$  is a real number. The weak formulations is derived in the standard way by multiplying the equation (1) by the test function  $\nu$ , than integrating over the domain  $\Omega$ , and applying the Green’s theorem (itegration by parts) to the second-order derivatives. Due to the homogeneous Dirichlet boundary conditions (2), the suitable solution space is  $V = H_0^1(\Omega)$ . The weak formulations is the following: find  $u \in V$  to satisfy the condition:

$$\int_{\Omega} \nabla u \cdot \nabla \nu dx = CONST_F \int_{\Omega} \nu dx \quad \forall \nu \in V \quad (3)$$

---

**Remark:** The folloing code is useable both in **FEMhub** by importing it from the .py file and in the **Online Lab**. The differences are in the importing from the file and plotting the results.

---

The first step is to import the necessary modules:

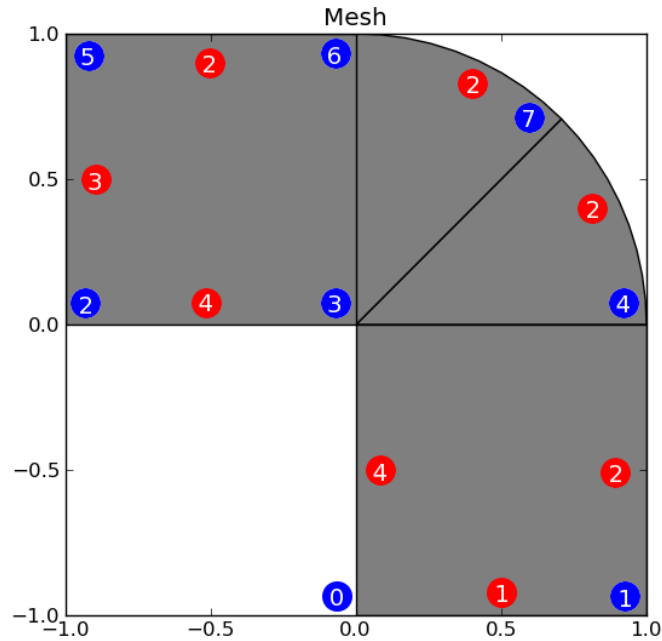


Figure 1: Initial mesh of the example problem.

```
# import the necessary modules
from hermes2d import Mesh, MeshView, H1Shapeset,
    PrecalcShapeset, H1Space, \
    WeakForm, Solution, ScalarView, LinSystem, DummySolver
from hermes2d.forms import set_forms
from hermes2d.examples.c03 import set_bc
```

Next create the variable for the mesh and create/import predefined mesh:

```
mesh=Mesh()
# importing the mesh from the file
mesh.load_hermes("poisson-L.mesh")
```

Let see the initial mesh (Fig 1):

```
# plotting the mesh
mview=MeshView("Mesh")
mview.show(mesh, lib="mpl", notebook=True,
    filename="meshLOD-init.png")
```

Refine the mesh (NB: it is valid only for the particular mesh of the present example):

```
# refining the mesh
mesh.refine_all_elements()
mesh.refine_towards_vertex(3,4)
mesh.refine_towards_boundary(2,4)
```

Let see the refined mesh (Fig. 2):

```
# plotting the mesh
```

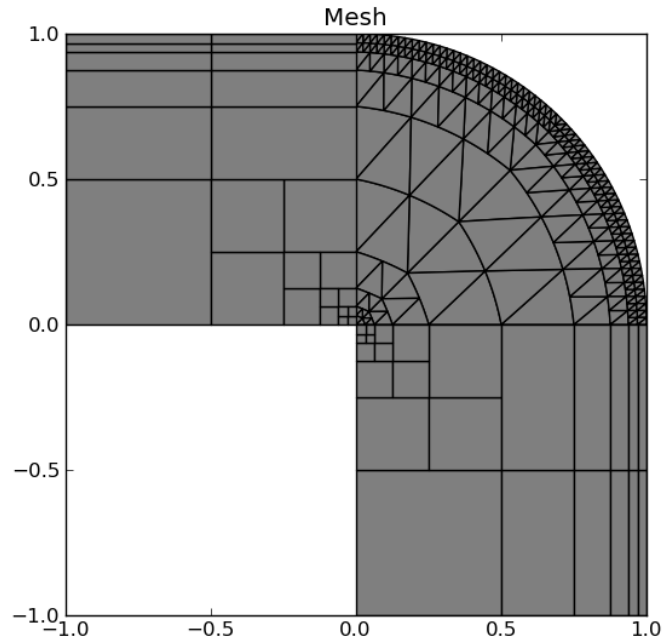


Figure 2: Refined mesh of the example problem.

```
mview.show(mesh, lib="mpl", notebook=True,
            filename="meshLOD-refined.png")
```

Define the shaperset:

```
shaperset = H1Shaperset()
pss = PrecalcShaperset(shaperset)
```

*PrecalcShaperset* stores the precalculated values of the shape functions. This is because **Hermes** saves all values of the polynomials of the shape functions to the cache during the matrix assembly to increase the performance.

Next define the space of the shape functions:

```
# creating H1 space
space=H1Space(mesh, shaperset)
space.set_uniform_order(2)
space.assign_dofs()
```

**H1Space** represent a space of the continuous scalars in the domain (mesh). The input arguments are pointers to the *mesh* and *shaperset*.

**set\_uniform\_order** sets the uniform polynomial range for all elements as a function input argument.

**assign\_dofs** creates the base functions and assigns them the degree of freedom.

Defining the boundary conditions:

```
# boundary conditions
    set_bc(space)
```

The problem to solve is defined as following:

```
# initialising the weak form
    wf = WeakForm(1)
    set_forms(wf)
```

**WeakForm** class represents the weak formulation of the PDE and it requires the count of the equations in the system as an input argument, which is 1 in the present example.

If the weak formulation and the discretization determined by the space and the mesh are constructed, then the problem will be approximated by the Galerkin method. This method is the heart of the *Hermes* and makes possible to create a disperse system of the linear equations represented by the *LinSystem* class. The solution of the linear system leads to the approximated solution of the original problem.

*LinSystem* class needs three components: weak formulation, function space and an external solver like CG or UMFPACK. The following code creates a linear solver, initialises the *LinSystem* class and passes the pointer to previously defined *HISpace*:

```
# initialising the linear system and the solver
    solver = DummySolver()
    sys = LinSystem(wf, solver)
    sys.set_spaces(space)
    sys.set_pss(pss)
```

Finally, the *LinSystem* assembles the stiffness matrix and the right hand side and solves the linear system:

```
# assembling the stiffness matrix and solving the system
    sln = Solution()
    sys.assemble()
    sys.solve_system(sln)
```

At this point the Poisson problem is solved. The last two rows in the code can be repeated during the solving of the time-dependent problem as many times as needed. The *Solution* class contains the approximated solution of the PDE right after the *LinSystem::solve()* has finished, and this solution can be visualised by the *ScalarView* class (Fig. 3). Also the ranks of the mesh elements can be plotted (Fig. 4):

```
# visualisation of the solution
    view = ScalarView("Solution")
    view.show(sln, lib="mayavi", filename="poisson.png",
              notebook=True)
# plotting the mesh
    mview.show(mesh, space=space, lib="mpl", notebook=True,
               filename="meshLOD-space.png")
```

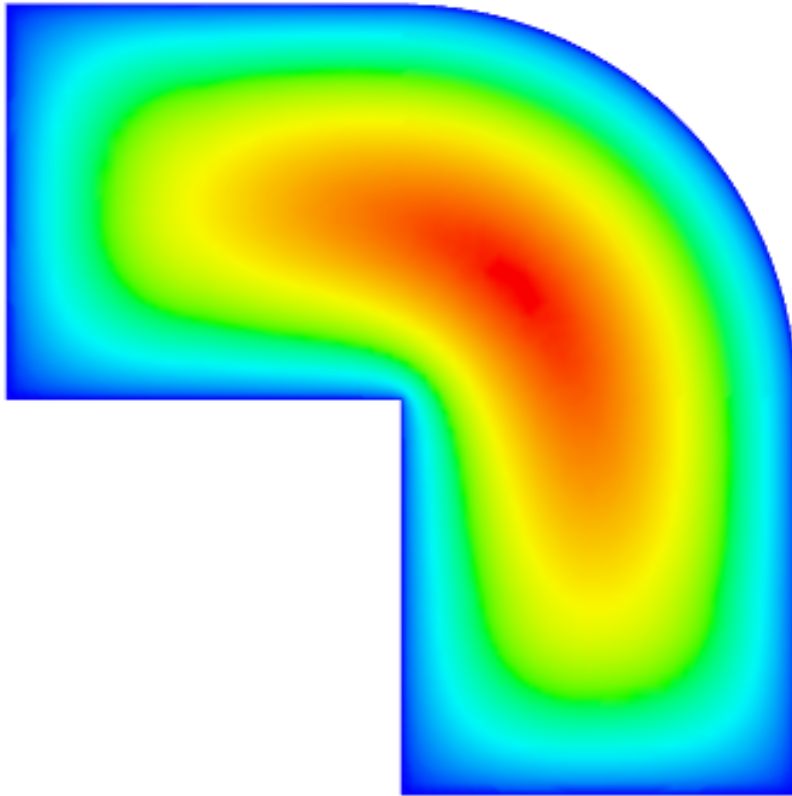


Figure 3: The solution of the example problem.

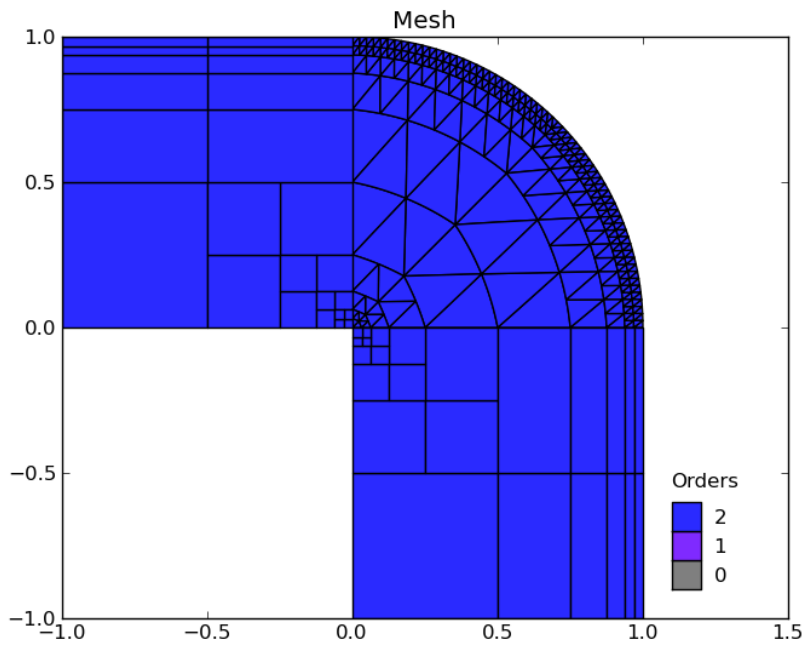


Figure 4: The ranks of the mesh elements of the example problem.



---

**Task 1:** Creating the Example Mesh.

Create mesh after Fig. 1 . Plot the mesh and the element ranks and show the code what creates this mesh.

---

---

**Task 2:** Non-homogeneous (Non-zero) Dirichlet Boundary Bonditions.

The Poisson problem (1) , (2) is modified as:

$$-\Delta u(x, y) = CONST_F, \quad u(x, y) = -\frac{CONST_F}{4}(x^2 + y^2) \quad in \quad \delta\Omega \quad (4)$$

The analytical solution of this problem is

$$-\frac{CONST_F}{4}(x^2 + y^2) \quad (5)$$

Since the Dirichlet boundary conditions match the functions  $u(x, y)$  then this functions is an exact solution. The remark is that since the exact solution is a polynomial, **Hermes** calculates it exactly while the all mesh elements are quadratic or higher-order elements. If any of the elements is a linear element, only the approximation will be found.

- 1 Make the following modifications in the code of the example problem above:
    - (a) import the boundary conditions from the example *c04* of the **hermes2d**;
    - (b) keep the mesh the same;
    - (c) replace the existing refinement of the mesh by the double refinement over all elements. It can be done by the *for-cycle*;
    - (d) set the rank of the elements to **2**;
    - (e) add **-4** as a second argument for the *set\_forms*.
  - 2 Solve the problem, plot the solution, refined mesh, the ranks of the elements, and show the code that solves the problem.
-

---

**Task 3: Neumann Boundary Bonditions.**

This time the problem is

$$\begin{aligned} -\Delta u &= CONST_F & u &= 0 & \text{on } \Gamma_4 \\ \frac{\delta u}{\delta n} &= C_1 & & \text{on } \Gamma_1 \\ \frac{\delta u}{\delta n} &= C_2 & & \text{on } \Gamma_2 \\ \frac{\delta u}{\delta n} &= C_3 & & \text{on } \Gamma_3 \end{aligned} \quad (6)$$

where  $\Gamma_1 \dots \Gamma_4$  correspond to the edges 1...4 in Fig. 1. The weak formulation contains some surface integrals:

$$\int_{\Omega} \nabla u \cdot \nabla v dx = CONST_F \int_{\Omega} v dx + C_1 \int_{\Gamma_1} v dl + C_2 \int_{\Gamma_2} v dl + C_3 \int_{\Gamma_3} v dl \quad (7)$$

In fact, in the *Hermes* all standard forms  $a(u, v) = l(v)$  are defined as the sum of the volume and surface integrals. In the case of the linear form  $l(v)$ , it means that

$$l(v) = \sum_m l_m^{vol}(v) + \sum_n l_n^{surf}(v) \quad (8)$$

**1** Make the following modifications in the code of the example problem above:

- (a) import the boundary conditions and the weak form from the example *c05* of the *hermes2d*;
- (b) import the weak form as a variable *set\_forms\_surf* from the same example;
- (c) set **4** as a ranks of the polynomials of all elements;
- (d) use the mesh created in the example problem;
- (e) replace the existing refinement by the refinement of the mesh **12** times towards the node **3**;
- (f) add **-1** as a second argument while initialising the weak frmulation;
- (g) add *set\_forms\_surf(wf)* as a weak form for the surface.

**2** Solve the problem, plot the solution, refined mesh, the ranks of the elements, and show the code that solves the problem.

---

---

**Task 4:** Newton Boundary Conditions.

The Newton boundary conditions has a formulation:

$$\frac{\delta u}{\delta n} + c_1 u = c_2, \quad c_1 \neq 0 \quad (9)$$

The Laplace equations describes the stationary heat conductivity in the homogeneous linear material among the other phenomena. Similarly to the Neumann conditions, the Newton conditions also contain surface integrals.

- 1 Make the following modifications in the code of the example problem above:
  - (a) import the boundary conditions and the weak form from the example *c06* of the *hermes2d*;
  - (b) set **3** as a rank of the polynomials of all elements;
  - (c) replace the existing refinement by the double refinement of all elements and then refine towards the node **3 12** times.
- 2 Solve the problem, plot the solution, refined mesh, the ranks of the elements, and show the code that solves the problem.

---

**Task 5:** General 2nd Order Linear Equation.

The present example solves the general 2nd order linear equation with non-constant coefficients. The form of the 2nd order linear equation is

$$\begin{aligned} & -\frac{\delta}{\delta x}(a_{11}(x, y)\frac{\delta u}{\delta x}) - \frac{\delta}{\delta x}(a_{12}(x, y)\frac{\delta u}{\delta y}) - \frac{\delta}{\delta y}(a_{21}(x, y)\frac{\delta u}{\delta x}) - \frac{\delta}{\delta y}(a_{22}(x, y)\frac{\delta u}{\delta y}) + \dots \\ & \dots + a_1(x, y)\frac{\delta u}{\delta x} + a_{21}(x, y)\frac{\delta u}{\delta y} + a_0(x, y)u = rhs(x, y) \end{aligned} \quad (10)$$

with Dirichlet and/or Neumann boundary conditions. *rhs* is the right hand side of the equation.

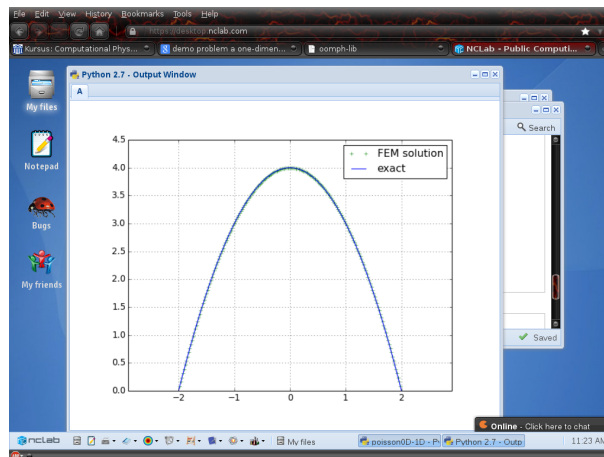
- 1 Create an initial mesh consisting of a square.
- 2 Make the following modifications in the code of the example problem above:
  - (a) import the boundary conditions and the weak form from the example *c07* of the *hermes2d*;
  - (b) set **2** as a rank of the polynomials of all elements;
  - (c) replace the existing refinement by the refinement of **4** times over all elements.
- 3 Solve the problem, plot the solution, refined mesh, the ranks of the elements, and show the code that solves the problem.

# Computational Physics II

## Practical Work

### FEMhub

### Programming



Heiki Kasemägi

October 3, 2013

# Contents

- Programming** **3**
- Introduction . . . . . 3
- Passing the Practical Work . . . . . 3
- References . . . . . 3
- Example problem: 1D Poisson Equation with homogeneous (zero) Dirichlet Boundary Conditions . . . . . 4

# List of Figures

1	The solution of the Poisson equation. . . . .	4
---	-----------------------------------------------	---

# Programming

## Introduction

The following example creates a program in the *FEMhub Online Lab* what solves the 1D problem with the linear elements.

## Passing the Practical Work

To successfully pass the present practical work, it is necessary to submit a report containing the correct solutions to the tasks marked by double lining and the heading “**Task #**”. The solution should contain correct task setup, solution, explanation of the solution, symbols and notations. There are no restrictions to include auxiliary material into the report. The student submitting the report should be ready to explain the report in oral and/or written form if necessary. There may rise the need to improve the solution and/or renew or complement the report. The deadline of the submission is the next Midnight 2 weeks (14 calendar days) after the scheduled Practical Work. E.g., if the Practical Work takes place Sept. 2, the deadline is 00:00am Sept. 17. The time is localtime. The report should be in the correct form, including the title page containing the information about the author, the name of the Practical Work etc. The accepted file format is PDF (Portable Document Format). The report should be a single file accompanied always by the simulation files. The plots, pictures, graphs etc. should have readable font size, title(s), legend(s) etc. The report and accompanying files are submitted via moodle.

## References

The present practical work bases on the “Demo Problem: A One-Dimensional Poisson Problem” [http://oomph-lib.maths.man.ac.uk/doc/poisson/one\\_d\\_poisson/html/index.html](http://oomph-lib.maths.man.ac.uk/doc/poisson/one_d_poisson/html/index.html) in “The Object-Oriented Multiphysics Finite Element Library (OOMPH-LIB)” <http://oomph-lib.maths.man.ac.uk/doc/html/index.html>.

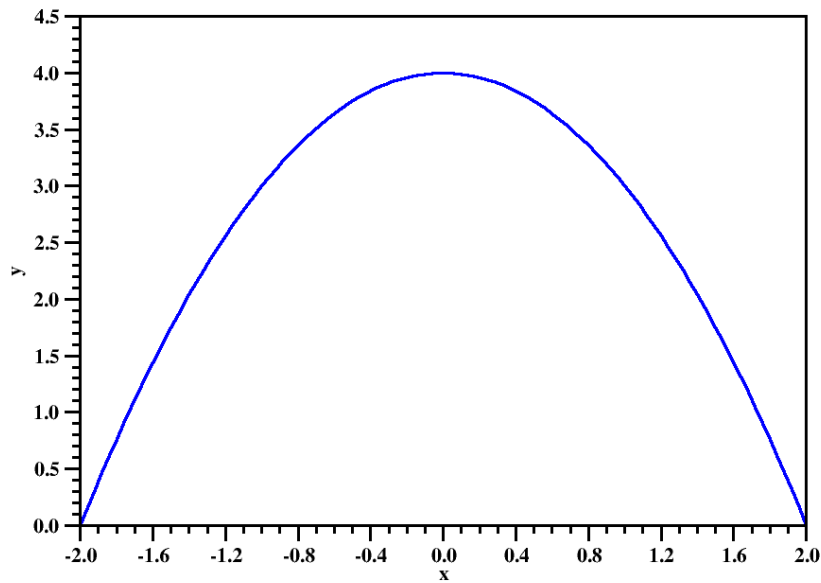


Figure 1: The solution of the Poisson equation.

## Example problem: 1D Poisson Equation with homogeneous (zero) Dirichlet Boundary Conditions

The equations to be solved is

$$-\Delta u(x) = f(x), \quad u(x) = 0 \quad \text{on} \quad \delta\Omega, \quad f(x) = \text{const} \quad (1)$$

The analytical solution of this differential equation, with the boundary conditions

$$f(x) = 2, \quad u(-2) = 0, \quad u(2) = 0 \quad (2)$$

is plotted as a Fig. 1 .

To solve the equation (1) with the boundar conditions (2) by the Finite Element method, the following code is written. The code can be run from the command line of the **FEMhub** or preferrably in the **Online Lab** of the **FEMhub**.

First, import the necessary modules and functions:

```
# Import symbolics , plotting , and linear algebra functions .
from pylab import plot , savefig , grid , legend , clf , pcolor ,
    spy , axis
from numpy import arange , zeros , dot , array , sin , cos , log ,
    exp , tan
from numpy.linalg import norm
from sympy import var , pi , factorial , lambdify
from scipy import shape , eye , tril , triu , diag , sparse
```



```

from scipy.linalg import solve
from scipy.sparse.linalg import cg, cgs, qmr, gmres, bicg,
    bicgstab
from scipy.sparse.linalg.dsolve import spsolve
from time import time

```

Define the function what creates a 1D mesh:

```

def create_1Dmesh(xmin=-101,xmax=101,step=0.02):
    newmesh=[]
    for i in range(xmin,xmax):
        newmesh.append(i*step)
    return newmesh

```

---

**Remark:** *range(x,y)* outputs values  $z$  in the range

$$x \leq z < y$$


---

Next, start with the main function.

```

def solve_poisson0D1D(mesh, f_const=2.):

    # the number of the elements
    n_elem = len(mesh) - 1
    # the size of the matrix
    size = n_elem - 1

    # creating empty sparse matrixes
    # I, J are the indexes of the system matrixes, V is the
    # values determined by # those indexes
    I = []
    J = []
    V = []
    # right hand vector
    rhs = zeros(size)
    # assembling the matrix and right hand vector (the cycle
    # over the elements)
    print "assembling ..."

    # the system matrix is created with a for-cycle
    # the given for-cycle contains a discrete Poisson equation
    # in its weak form
    for i in range(0, n_elem):
        x1 = mesh[i]
        x2 = mesh[i+1]
        h = float(x2 - x1)
        # boundary elements

```

```

if i == 0: # the first boundary condition
    I.append(i); J.append(i); V.append(1./h)
    rhs[i] = rhs[i] + h*f_const/2.
elif i == n_elem-1: # the second boundary condition
    I.append(i-1); J.append(i-1); V.append(1./h)
    rhs[i-1] = rhs[i-1] + h*f_const/2
else:
    I.append(i-1); J.append(i-1); V.append(1./h)
    I.append(i); J.append(i); V.append(1./h)
    I.append(i); J.append(i-1); V.append(-1/h)
    I.append(i-1); J.append(i); V.append(-1/h)
    rhs[i-1] = rhs[i-1] + h*f_const/2.
    rhs[i] = rhs[i] + h*f_const/2.
# solving the matrix equation
print "solving ..."
m = sparse.coo_matrix((V,(I,J)),shape=(size , size))

```

*sparse.coo\_matrix* creates a sparse matrix in the coordinate (**COO**) format, which has the coordinates of the matrix elements in the vector form and the third input vector consists of the values of the matrix elements; the matrix contains only non-zero elements. This matrix can not be used in the arithmetical operations.

```
m = m.tocsr()
```

*tocsr* converts the **COO** matrix into the **CSR** (*Compressed Sparse Column*) matrix. The arithmetical operations are more effective using the latter format.

```

rhs = array(rhs)
#sol, res = gmres(m, rhs, tol=1.e-8) # more solvers to
use: cg, cgs, qmr, gmres, bicg, bicgstab, ...
sol = spsolve(m, rhs)

```

*spsolve(A,b)* solves the sparse linear system  $Ax = b$  with the SuperLU as a default solver, which is replaced automatically upon the first chance by the UMFPACK solver. The commented row shows, how to use the other available solvers.

```

# plotting the solution
x_array = array(mesh)
sol_ext = array([0] + list(sol) + [0])
print "plotting ..."
clf()
axis('equal')
label = "solution"
grid(True)
plot(mesh, sol_ext, "g-", label=label)
legend()
lab.show()

```

In order to solve the system the mesh generator and the function solving the system need to be called:

```

mesh=create_1Dmesh()
solve_poisson0D1D(mesh,2.)

```

If the compactness of the code is desired, the mesh generator can be moved to the body of the equation solver and the headers of the both functions have to be modified accordingly.

---

### **Task 1:**

The aim of the tasks is to extend and complement the example code.

- 1** Derive the analytical solution of the differential equation (1) with boundary conditions (2) and show the solution procedure along with the solution.
  - 2** Modify the function *solve\_poisson0D1D* to calculate and plot the exact analytical solution of the equation (1) with boundary conditions (2) along with the finite element solution. Plot the solution and show the code.
  - 3** Modify the code to calculate the relative error of the finite element solution. Show relative error and the code.
  - 4** Write a function to calculate and plot the dependence of the relative error on the mesh resolution. Plot the results and show the code.
- 

### **Task 2:**

The equation to solve is (1) with the boundary conditions

$$f(x) = 2, \quad u(0) = 0, \quad u(1) = 0 \quad (3)$$

- 1** Derive the analytical solution of this equation with given boundary conditions and show the solution procedure along with the solution.
  - 2** Write a program runnable within the *FEMhub* environment to solve this problem using the Finite Element Method. Plot the Finite Element solution and the exact solution in the same plot and show the code of the program.
  - 3** Calculate and plot the dependence of the relative error on the mesh resolution. Show the program code.
-

---

**Task 3:**

The equation to solve is (1) with the boundary conditions

$$f(x) = 30\sin(\sqrt{30}x), \quad u(0) = 0, \quad u(1) = -1 \quad (4)$$

- 1** Derive the analytical solution of this equation with given boundary conditions and show the solution procedure along with the solution.
  - 2** Write a program runnable within the *FEMhub* environment to solve this problem using Finite Element Method. Plot the Finite Element solution and the exact solution in the same plot and show the code of the program.
  - 3** Calculate and plot the dependence of the relative error on the mesh resolution. Show the program code.
-