

Elmer FEM

open source multiphysical simulation software

Elmer GUI Tutorials

CSC – IT Center for Science

April 6, 2023

ElmerGUI Tutorials

About this document

The Elmer GUI Tutorials is part of the documentation of Elmer finite element software. Elmer GUI Tutorials gives examples on the use of Elmer in different fields of continuum physics. Also coupled problems are included.

All these tutorials assume the use of ElmerGUI, the graphical user interface of Elmer. There are also older tutorials in the Elmer non-GUI Tutorials that may be useful for power users who want to understand more deeply how models in Elmer are set up.

The present manual corresponds to Elmer software version 9.0.

Latest documentations and program versions of Elmer are available (or links are provided) at <http://www.csc.fi/elmer>.

Copyright information

This document is licensed under the Creative Commons Attribution-NonCommercial 3.0 License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/>.

Initially these tutorials have been written by the Elmer team at CSC - IT Center for Science. However, Also external contributions to the tutorials are welcome.

Contributors

We gratefully acknowledge Rich Bayless for proofreading, correcting and updating the tutorials to be compatible with Elmer version 9.0.

Contents

Table of Contents	3
1 Heat equation – 3D – Temperature field of a solid object	7
2 Heat equation – 2D – Temperature field of an L-shaped domain	29
3 Heat Equation –1D – Temperature of an idealized geological intrusion	34
4 Heat Equation – 2D – Axi Symmetric Steady State Radiation	40
5 Heat Equation – 2D – Active and Passive elements	45
6 Linear elasticity equation – 2D – Loaded elastic beam	51
7 Linear elasticity equation – 3D – Loaded elastic beam	55
8 Non-linear elasticity equation – 3D – Loaded elastic curve	59
9 Smitc solver – 2D – Deflection of a linear elastic plate	64
10 Smitc solver – 2D – Eigenmodes of an elastic plate	69
11 Electrostatic equation – Computation of fringe capacitance	74
12 Electrostatic equation – Capacitance of two balls	79
13 Electrostatic equation – Capacitance of perforated plate	85
14 Magnetic field induced by harmonic current in a wire	90
15 Magnetostatics – Magnetic field resulting from a permanent magnet	97
16 Harmonic magnetic field – 2D – Induction heating of a graphite crucible	102
17 Helmholtz – 2D – Acoustic Waves – Air in a Cavity	107
18 VectorHelmholtz – model wave propagation in bent waveguide	112
19 Navier-Stokes equation – Laminar incompressible flow passing a step	118
20 Navier-Stokes equation – 2D – Incompressible – Driven Cavity	122
21 Navier-Stokes equation – Turbulent incompressible flow passing a step	136
22 Navier-Stokes equation – Laminar compressible flow over a step	141
23 Vortex shedding – von Karman instability	148

24 Thermal flow in curved pipe	152
25 Interaction between fluid flow and elastic obstacle	158
26 Flow and Heat –2D – Transient – Rayleigh-Benard instability	165
27 Temperature distribution of a toy glacier	170
28 Temperature and velocity distributions of a toy glacier and bedrock	176
29 Generic scalar PDE on 3D angle domain	181
30 Electro-kinetics – 2D – Electro-osmotic flow and advected species	186
31 Asymmetrical Conductor with a Hole TEAM Workshop Problem 7	193
Index	201

Instructions for the Elmer tutorials using ElmerGUI

Here are some general instructions for following the Elmer tutorials

- The Elmer tutorials consist of the tutorial documentation and the tutorial folders. For example, these ElmerGUI tutorials are described in the document 'ElmerTutorials.pdf' and the working project files are stored under the folder 'tutorials-GUI-files'.
- If you need a copy of the folder 'tutorials-GUI-files', it can be downloaded from: <https://www.nic.funet.fi/pub/sci/physics/elmer/doc/>
- Each sub-folder under 'tutorials-GUI-files' contains all the files needed to run a particular tutorial, such as the ElmerGUI project file, the geometry input file, and the generated mesh files. If you wish to just run the existing project tutorial and examine the resulting output, start ElmerGUI and load the existing project.
- If you are more interested in learning how to use ElmerGUI by constructing an ElmerGUI project from the beginning, the first step will be to create your own sub-folder, and copy in the geometry input file from the tutorial of interest. Then start ElmerGUI and create a new project, following the step by step instructions in each tutorial.
- Many examples of geometry input files should be available among the `ElmerGUI/samples` directory that should have come with the Elmer installation. Look under a subdirectory named after the suffix of the sample file, where the suffix indicates the type of geometry format.
- The instructions written in `verbatim` refer to operations with the GUI. Indentation means step in the menu hierarchy. The ElmerGUI instructions should not be mixed with the statements in the `.sif` command file.
- The menu structure for the default set of equations is located in directory `edf`, there are a few additional ones in directory `edf-extra`. These may be copied to the directory `edf` permanently, or may be appended to the menus while running the ElmerGUI.
- The default menu structure may differ from the configuration used when writing the tutorial. Hence the user is encouraged to check by herself whether the menu structure needed for a particular tutorial has been loaded into ElmerGUI.
- After having once defined the case in ElmerGUI you may go to the working directory and launch ElmerSolver from command-line. There you may manually edit the `.sif` file using a text editor to alter the parameters.
- Manual alteration to the `.sif` file will not be communicated to the ElmerGUI project. All edits or changes will be overwritten by ElmerGUI the next time the project is saved. Copy the manually altered `.sif` files to another folder or change the name of the `.sif` file, to prevent losing your edits.

- It is assumed that the default method for visualization uses `.vtu` format result files. `ElmerPost` that uses the `.ep` format is no longer available in most installations. Fortunately, most of the tutorials currently output `.vtu` format result files, so Paraview (or ElmerVTK) may be used for visualization.
- Since ElmerGUI v. 9.0 the ElmerVTK widget that comes built into ElmerGUI has been upgraded to use `.vtu` files. ElmerVTK may be an ideal visualization tool for most straight-forward demonstrations, since it is easy to use. More complicated applications, such as animation of transient studies, may require the use of Paraview. Regardless of which post processing visualization tool is used, the construction of an ElmerGUI project remains the same for both.
- These cases have been run a number of times but errors are still possible. Reporting them to `elmeradm@csc.fi`, for example, or posting in the Elmer Forum at <http://www.elmerfem.org/forum/> is greatly appreciated.

Tutorial 1

Heat equation – 3D – Temperature field of a solid object

Directory: TemperatureGeneric

Solvers: HeatSolver

Tools: ElmerGUI,netgen,OpenCascade

Dimensions: 3D, Steady-state

Author: Mikko Lyly, Peter Råback

Introduction

As the first tutorial in ElmerTutorials.pdf, the basic usage of ElmerGUI will be demonstrated. New users of Elmer are invited to use this tutorial to learn how to use ElmerGUI and Elmer. Geometry will be provided, since creating geometry can be a barrier to using Elmer. Basic steps for geometry creation, for Windows users and for Linux users, are covered in the document GetStartedElmer.pdf.

Case definition

This tutorial tries to demonstrate how to solve the heat equation for a generic 3D object. The solid object (see figure 1.1) is heated internally by a heat source.

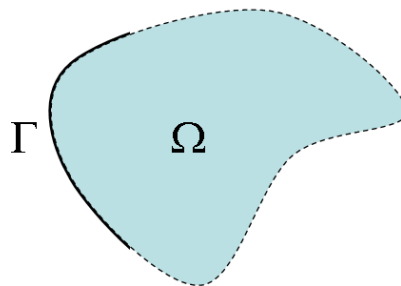


Figure 1.1: Generic object being heated

At some part of the boundary the temperature is fixed. Mathematically the problem is described by the Poisson equation

$$\begin{cases} -\kappa\Delta T & = \rho f & \text{in } \Omega \\ T & = 0 & \text{on } \Gamma \end{cases} \quad (1.1)$$

where κ is the heat conductivity, T is the temperature and f is the heat source. It is assumed that density and heat conductivity are constants.

To determine the problem we assume that part of the boundary is fixed at $T_0 = 293$ K, the internal heat generation is, $h = 0.01$ W/kg, and use the material properties of aluminium. ElmerGUI has predefined values for many constants and material properties defined with SI units, so we will use SI units throughout this tutorial. The goal of the simulation is to find out the temperature distribution in the object.

Existing Project

The Elmer tutorials consist of the tutorial documentation and the tutorial folders. For example, these ElmerGUI tutorials are described in the document 'ElmerTutorials.pdf' and the working project files are stored under the folder 'tutorials-GUI-files', and both can be downloaded from: <https://www.nic.funet.fi/pub/sci/physics/elmer/doc/>

Each sub-folder under 'tutorials-GUI-files' contains all the files needed to run a particular tutorial, such as the ElmerGUI project file, the geometry input file, and the generated mesh files.

If you wish to just run the existing project tutorial and examine the resulting output, start ElmerGUI and load the existing project file as follows:

```
File
  Load Project...
Run
  Start Solver
```

You should see the Convergence Monitor and the Solver log windows pop up, and the solution finish in a few moments in most cases. The results can then be examined using ElmerVTK or Paraview, as follow:

```
Run
  Start ElmerVTK
```

or

```
Run
  Start Paraview
```

New Project

If you are more interested in learning how to use ElmerGUI by constructing an ElmerGUI project from the beginning, then the next few sections will describe in detail the step by step actions needed to build an ElmerGUI project. The first step will be to create your own project folder, and copy in the geometry input file from the tutorial of interest. Then start ElmerGUI and create a new project, as follows:

```
Run
  New Project...
```

Start at the top and select the project directory as shown in Figure 31.3. Then select the Geometry input file, that was copied into your project folder. This could either be an elmer mesh or a geometry input file, such as an elmergrid input file. Lastly, if additional Equation Definition Files are needed for the tutorial, select as many as needed in the right hand box and add them to the left hand box. Finish by clicking on OK.

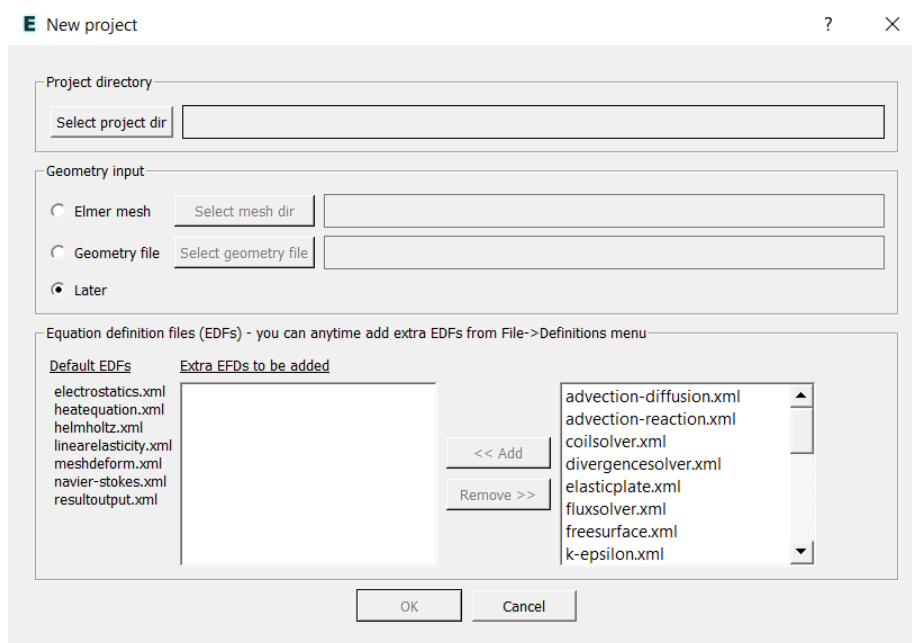


Figure 1.2: New Project

ElmerGUI Equation Definition Files

The New Project screen automatically locates and lists all of the available EDFs. The left side of the screen lists the Default EDFs, which will be loaded into each ElmerGUI project. The right hand box lists all of the extra EDFs, that are not normally loaded, allowing the option to add individual extra EDFs to a new project.

This tutorial will use the Default EDF, `heatequation.xml`, for the Heat Solver.

Solution procedure

Start ElmerGUI from command line or by clicking the icon in your desktop. Here we describe the essential steps in the ElmerGUI by writing out the clicking procedure. Tabulation (indentation) generally means that the selections are done within the window chosen at the higher level.

The geometry is given in step format in file `pump_carter_sup.stp` in the `samples/step` directory of ElmerGUI. This file is kindly provided at the AIM@SHAPE Shape Repository by INRIA. The heat equation is ideally suited for the finite element method and the solution may be found even at meshes that for some other problems would not be feasible. Therefore you may easily experiment solving the same problem with different meshes. If your particular version of ElmerGUI lacks OpenCascade, you might try to solve a similar problem with the `grd` files `angle3d.grd`, `angles3d.grd`, `bench.grd`, or `cooler.grd`, for example.

The CAD geometry defined by the step file is transformed on-the-fly by OpenCascade library into an `stl` file for which `nglib` creates tetrahedral volume discretization. You may also use the `tetlib` library (`tetgen`) if you have installed it as a plug-in.

Load the input file:

File

```
Open -> pump_carter_sup.stp
```

The meshing will take a minute or two. You should obtain your mesh and may check for the number of elements in the `Model` summary, as shown in Figure 1.3.

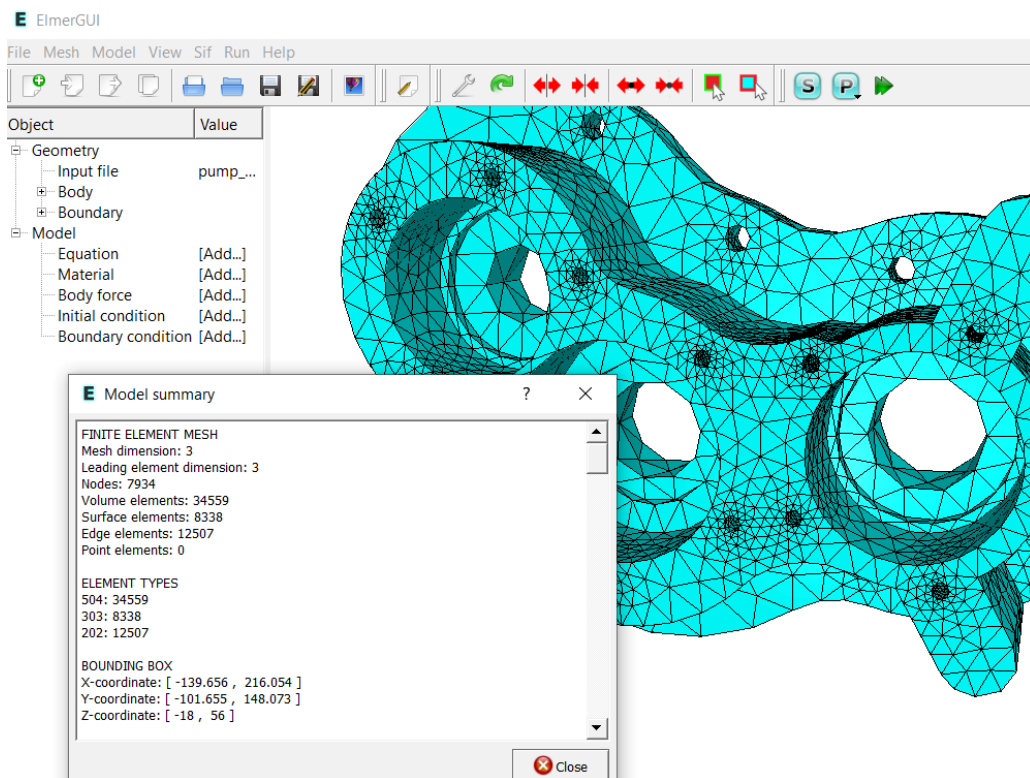


Figure 1.3: The computational mesh based on default settings

Visual inspection reveals that the mesh is not quite satisfactory in geometric accuracy. We choose to modify the mesh by altering the settings in the following way, as shown in Figure 1.4. In order to affect the mesh density we set the maximum element size, max h , and the minimum element size, min h , and select the option to restrict mesh size on surfaces by the STL surface density. Note that one must click on **Mesh**, **Remesh** in order for the new settings to be applied.

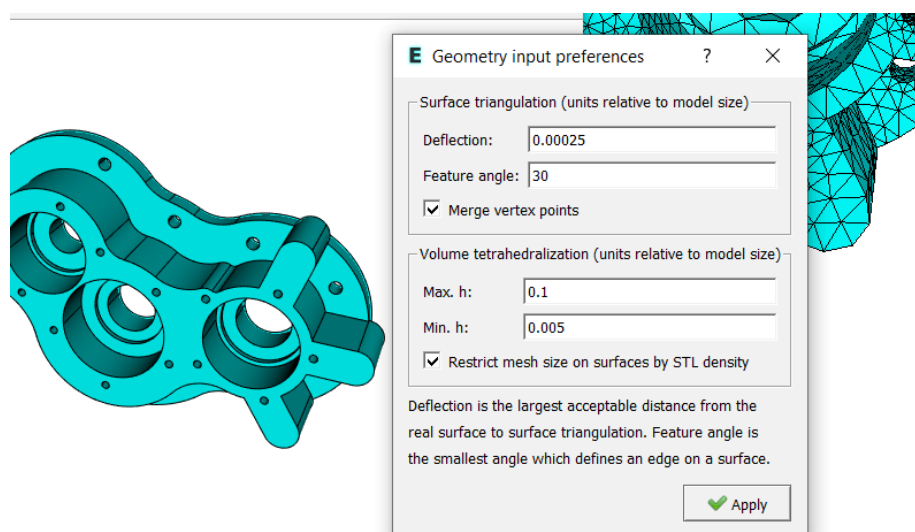


Figure 1.4: Refinement settings

```
View -> Cad model...
Model -> Preferences...
    Restrict mesh size on surfaces by STL density = on
    Apply
Mesh -> Remesh
```

The meshing operation takes a minute or two. The modified mesh should be more appealing to the eye. For example, notice the improvement in the inside diameters of the bores, they are more circular in shape. You may check for the number of elements in the Model summary, as shown in Figure 1.5.

Note that the remeshing operation will change the Elmer mesh.* files, but will not change the original geometry input file. The original step CAD file is unchanged, so any time the original geometry input file is input or reread, the meshing operations must be repeated.

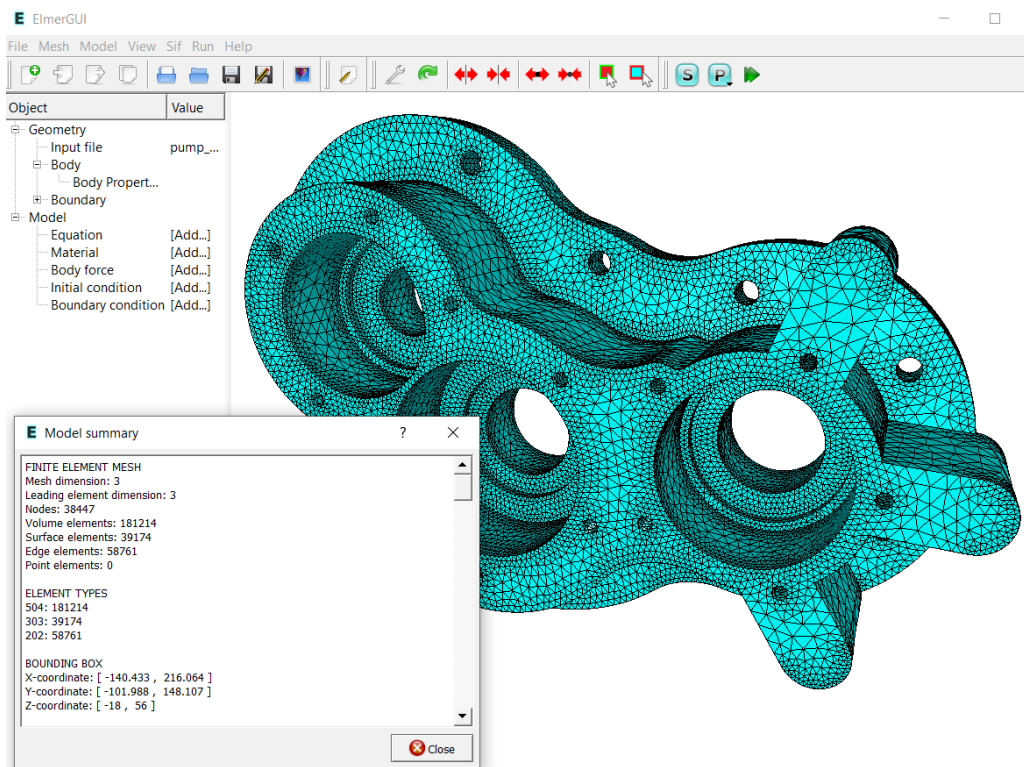


Figure 1.5: Refined mesh

We want to set the temperature at the inside of the three bores, each of which have two interior surfaces. We could set the boundary conditions by selecting all six surfaces, one at a time, as shown in Figure 1.6.

ElmerGUI has the capability to join boundary surfaces, so that we can unite all six boundaries into a single boundary, which will make it easier to apply our boundary conditions for the simulation. We demonstrate this optional step of 'Unify Surface' next. Note that the Elmer mesh.* files will be updated with the changed boundary surfaces, and the original geometry input file will be unchanged.

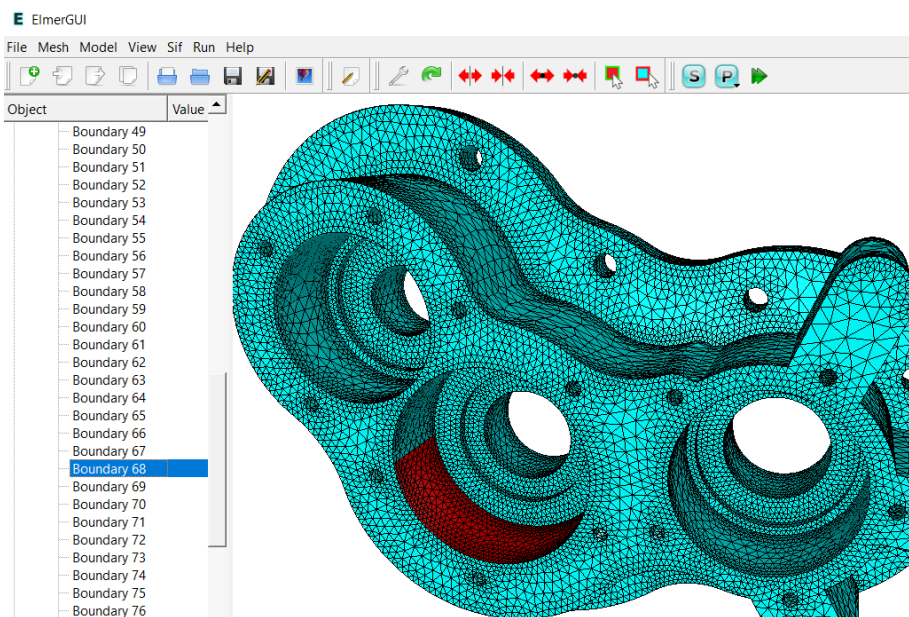


Figure 1.6: The computational mesh selecting one of the six interior boundaries

In order for ElmerGUI to unite surface boundaries, we must select the six pieces that constitute the boundaries. Select the first surface by double clicking on the surface, then add the next five surfaces to the selection by double clicking while holding down the `Ctrl`-key. Once all six surfaces are highlighted, use the menus to unify the surfaces.

Mesh
Unify Surface

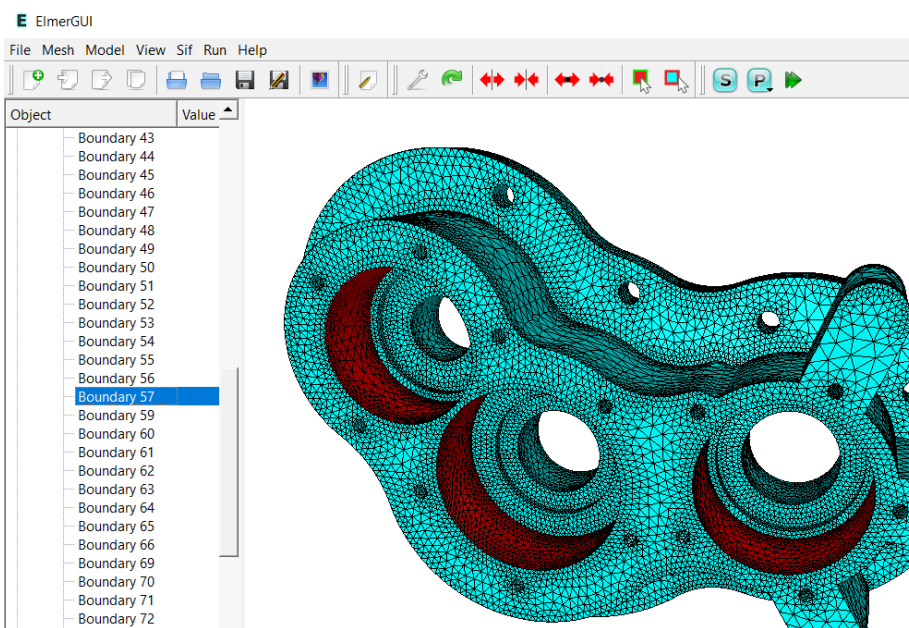


Figure 1.7: The computational mesh after uniting the six interior boundaries

After we have the mesh we start to go through the Model menu from the top to bottom. The EDF menus include default settings for each entry, and we will highlight which particular entries should be set or changed.

In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried out in 3-dimensional Cartesian coordinates and in steady-state. Only one steady-state iteration is needed as the case is linear.

For this particular case, we can accept all of the default settings under Setup. While the Setup window is open, take a look at all of the possible entries, as shown in Figure 1.8. Note the two free text input boxes, one at the top and the other at the bottom. Use the free text input boxes to add entries to the sif file such as comments about the model, or define constants, enter MATC commands, etc.

Model

Setup

Simulation Type = Steady state

Steady state max. iter = 1

Choose Apply to close the window.

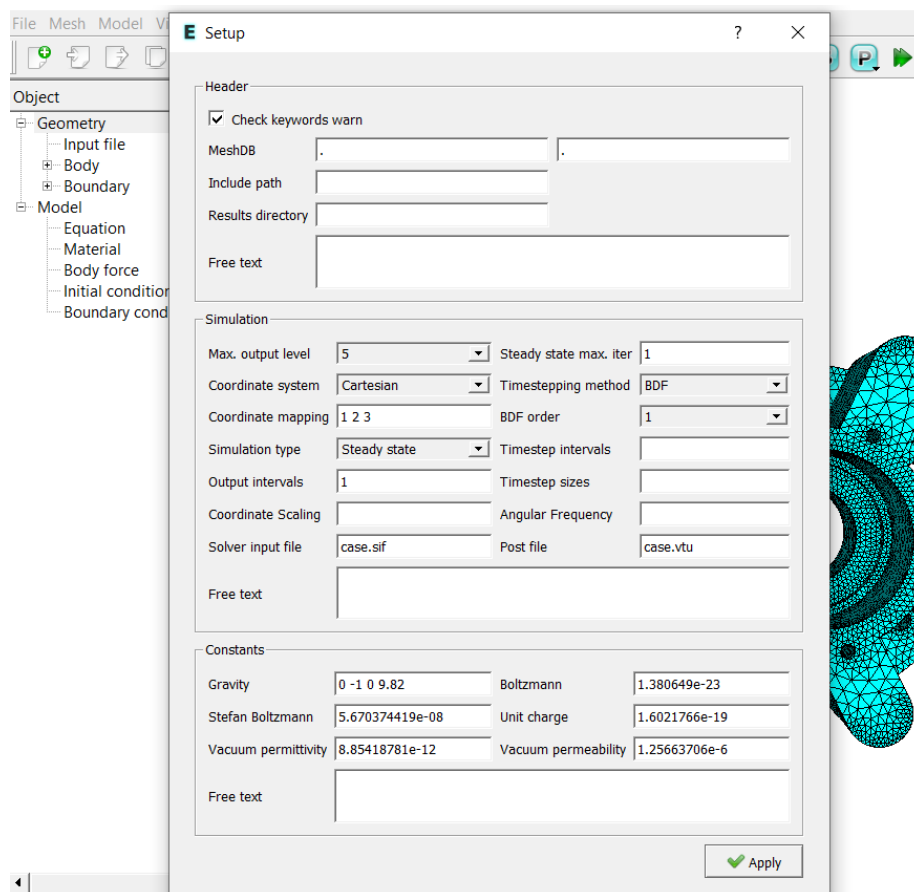


Figure 1.8: The Setup window

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have one set only one equation – the heat equation, as shown in Figure 1.9.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and therefore it is easier to assign the Equation and Material to the body directly, whereas the active boundary is chosen graphically.

```

Model
  Equation
    Add
      Heat Equation
        Active = on
        Apply to bodies = Body 1
        Name = Heat Equation
      Add
    OK

```

For the linear system solvers we are happy to use the defaults. Click on **Edit Solver Settings** to review the possible settings. One may however, try out different preconditioners (ILU1,...), for example.

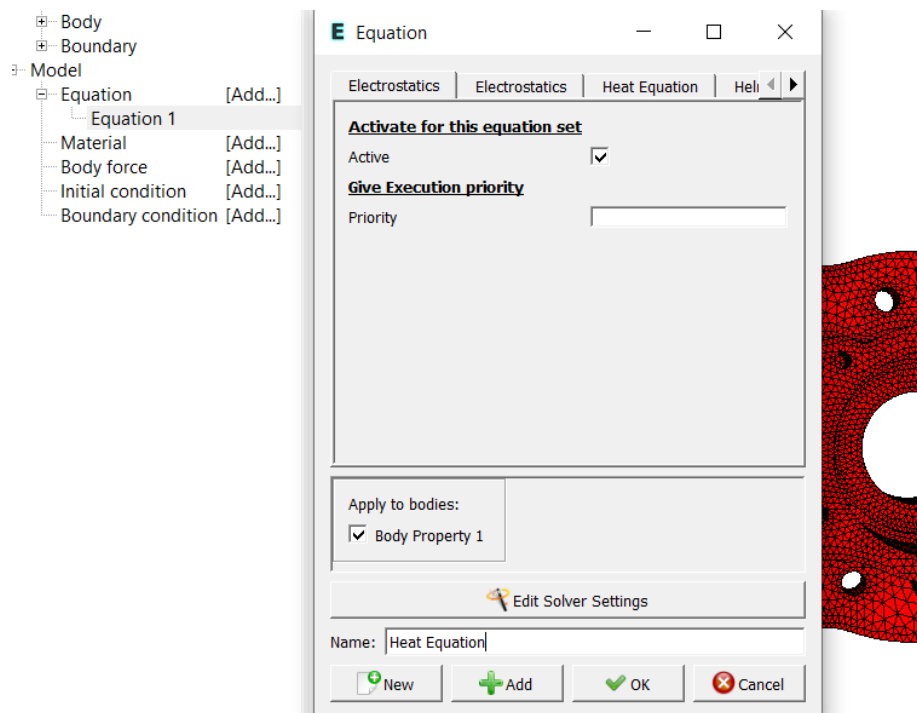


Figure 1.9: The Equation window

The Material section includes all the material parameters. The tabs near the top for each equation are divided to generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such as heat conductivity. We choose Aluminium from the Material library which automatically sets the needed material properties.

```

Model
  Material
    Add
      Apply to bodies = Body 1
      Material library
        Aluminium (generic)
      Add
    OK

```

Click on the tabs near the top of the window, such as 'General' or 'Heat Equation', to inspect the values set by selecting the predefined material from the Material Library, as shown in Figure 1.10.

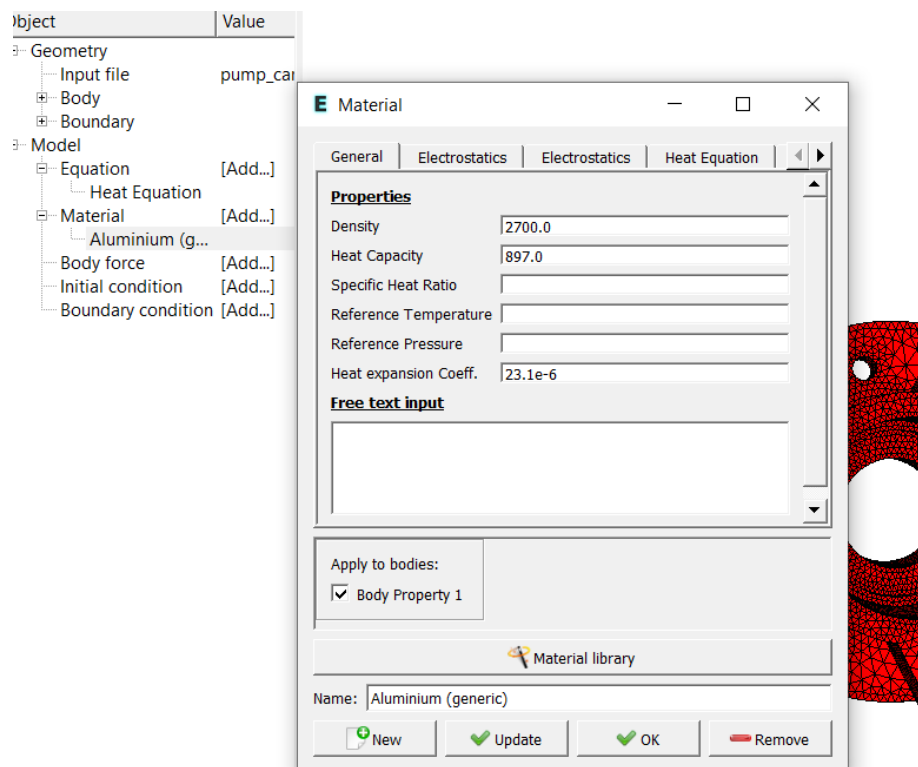


Figure 1.10: The Material window, showing the General tab

A Body Force represents the right-hand-side of a equation that in this case represents the heat source.

```

Model
  Body Force
    Add
      Heat Equation
        Heat Source = 0.01
        Apply to bodies = Body 1
        Name = Heating
      Add
    OK
  
```

Click on the tabs near the top of the window, to show the 'Heat Equation' tab, and inspect the possible values, as shown in Figure 1.11.

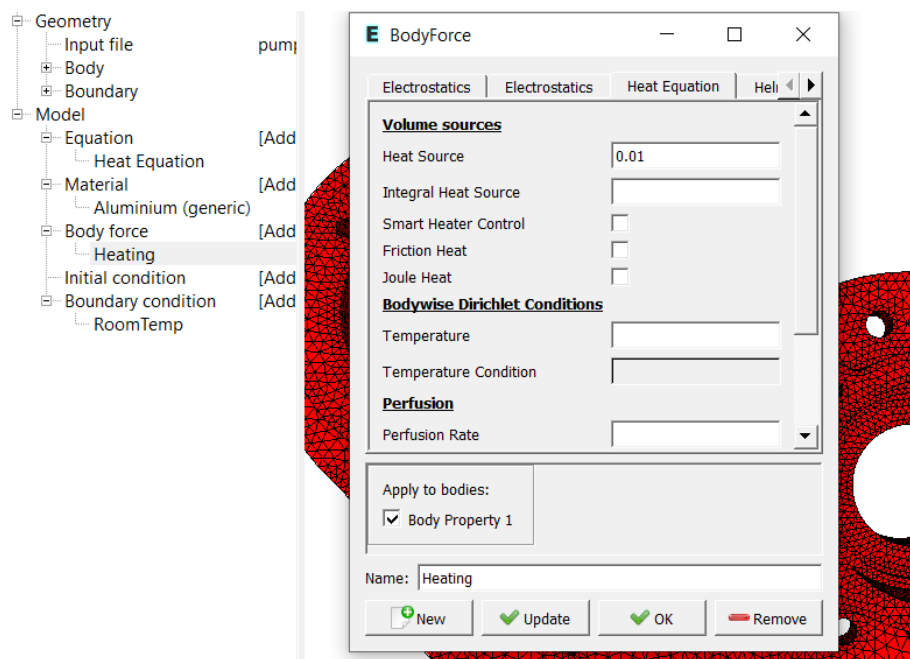


Figure 1.11: The Body Force window, showing the Heat Equation tab

No initial conditions are required in a steady state case. Initial conditions are usually only needed for transient studies. Note that the default temperature setting for any model is zero, which doesn't affect a steady state study, but will be important for a transient study. The Initial Condition input window is similar to the Body Force window, as shown in Figure 1.12.

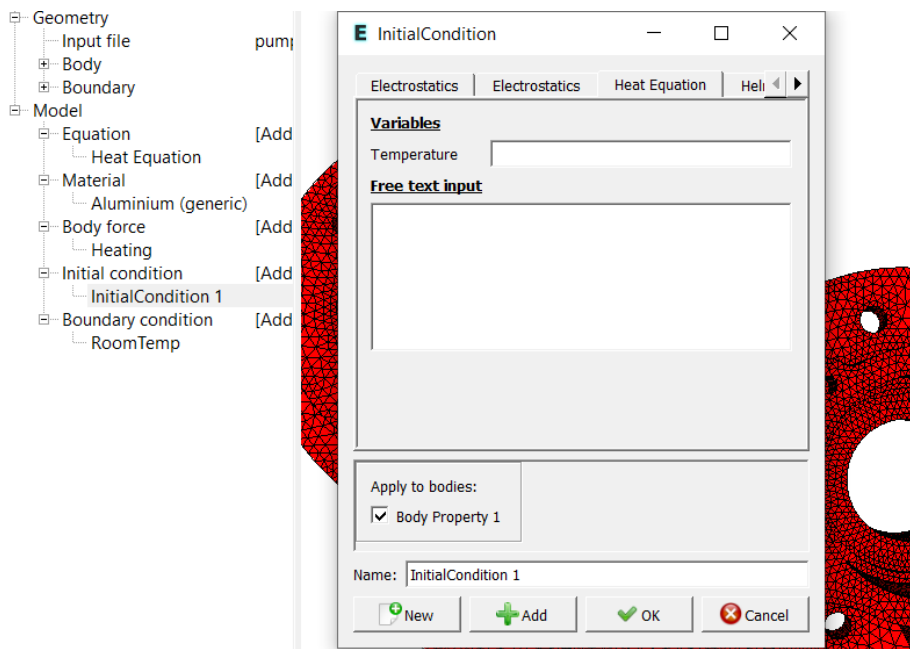


Figure 1.12: The Initial Condition window, showing the Heat Equation tab

In this case we will set only one boundary condition, which will be a fixed surface temperature, set to room temperature. You may have noticed that there are many surface boundaries listed for the geometry, as previously shown in Figure 1.6. For the heat equation, the default boundary condition is zero heat flux, or fully insulated, and for this example case we don't need to create boundary conditions for the other surfaces.

First we create the boundary condition, as shown in Figure 1.13. Note that we do not assign the new boundary condition to any particular boundary, that assignment will happen when we set the boundary properties.

```
Model
  BoundaryCondition
    Add
      Heat Equation
        Temperature = 293.0
      Name = RoomTemp
    Add
  OK
```

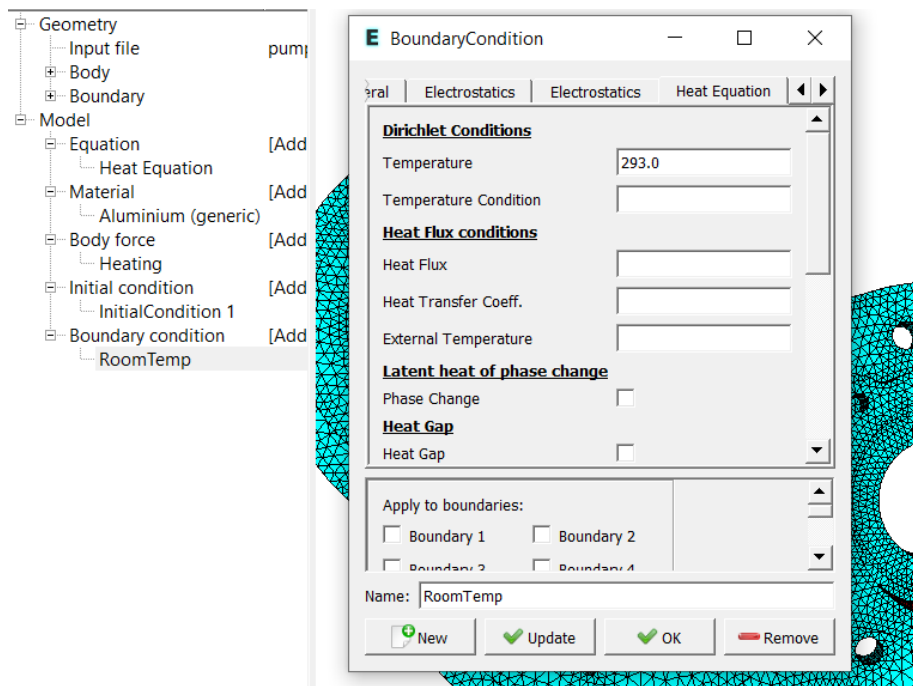


Figure 1.13: The Boundary condition window

Then we set the boundary properties, as shown in Figure 1.14. Select our desired boundaries and then assign a boundary condition to those boundaries.

```
Model
  Set boundary properties
```

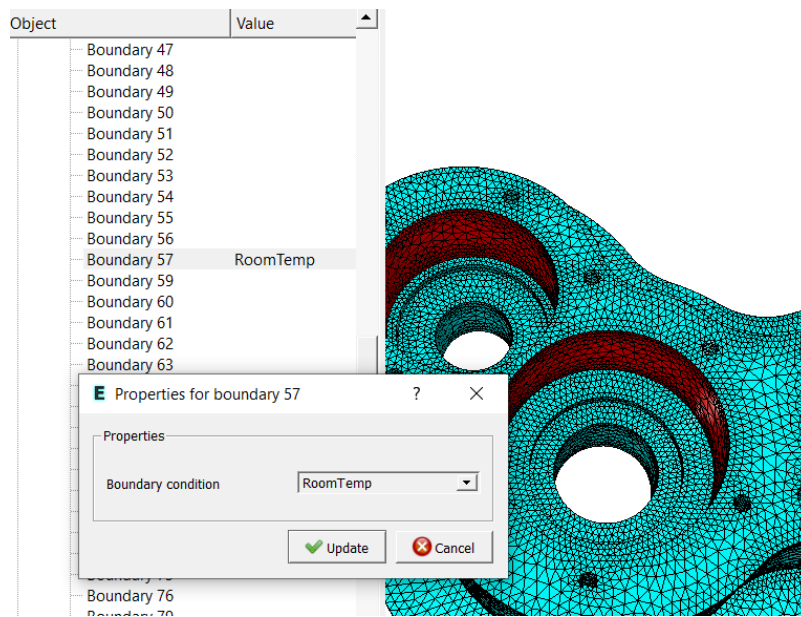


Figure 1.14: The Boundary property window

For our case, we will select the united group of three interior bores by double clicking with the mouse and then apply the condition for this boundary.

```
Boundary condition
RoomTemp
```

For the execution of the simulation, ElmerSolver needs the mesh files and the sif command file. We have now basically defined all the information for ElmerGUI to write the command file.

```
Sif
Generate
Edit -> look how your command file came out
```

After writing the sif file, we may also visually inspect, and optionally edit, the command file, as shown in Figure 1.15.

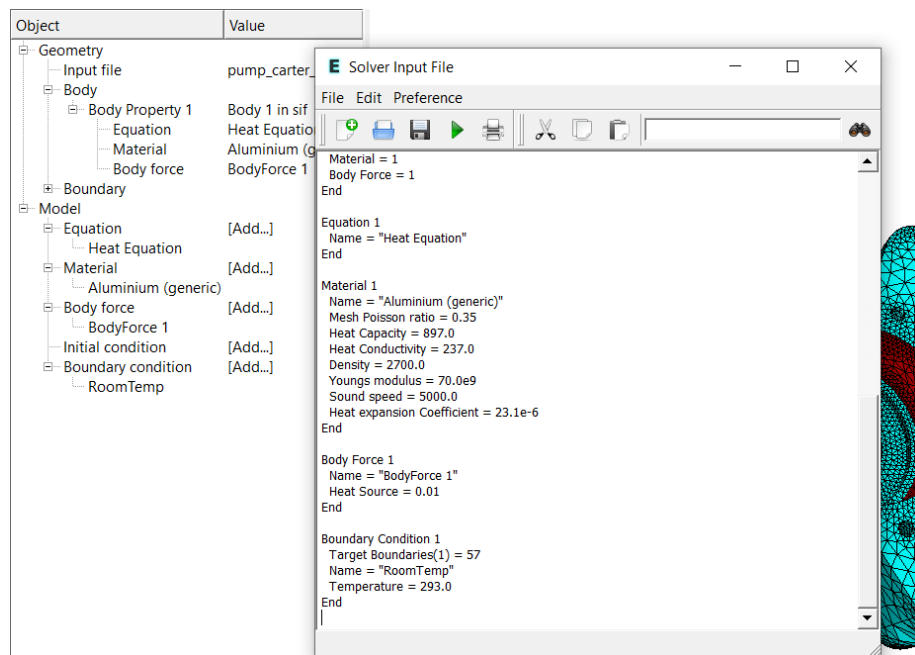


Figure 1.15: The Sif editor window

Before we can execute the solver we should save the files in a directory. In saving the project all the necessary files for restarting the case will be saved to the destination directory.

File

Save Project

After we have successfully saved the files we may start the solver

Run

Start solver

The solver log and a convergence view automatically pops up showing relative changes of each iteration, as shown in Figure 1.16.

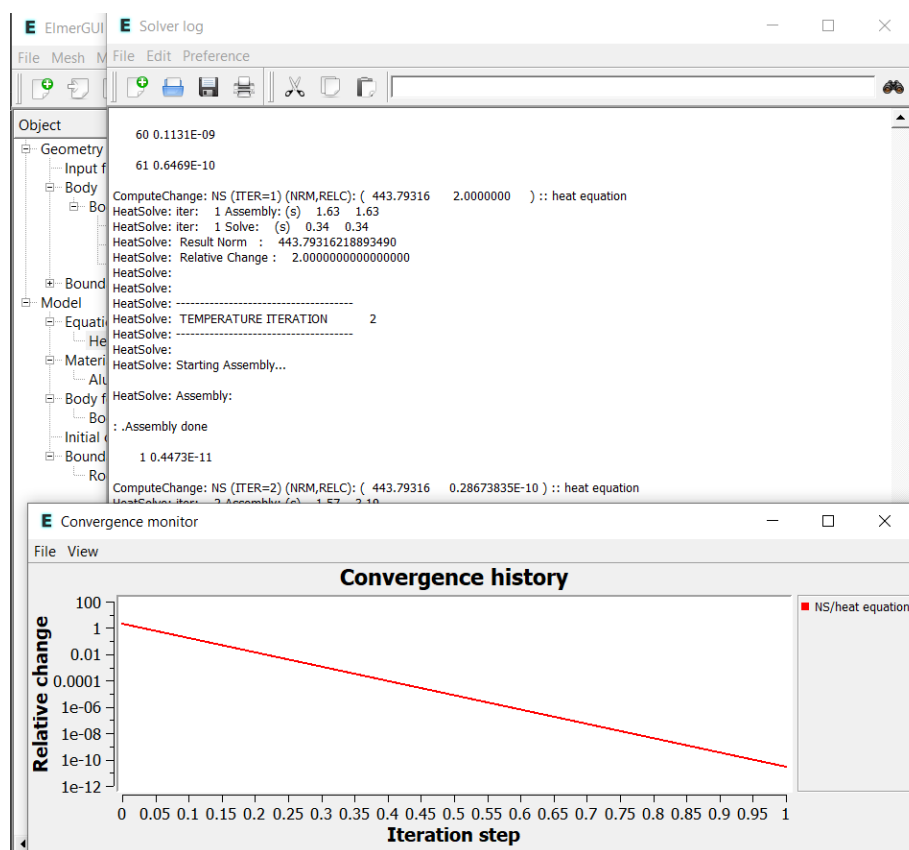


Figure 1.16: The Solver log and convergence windows

As the case is linear only one iteration was required for the solution and the second one just is needed to check the convergence. The norm of the solution should be around 432.4 K (with the default tetgen mesh 389.8 K, respectively).

Note: if you face problems in the solution phase and need to edit the settings, always remember to regenerate the `sif` file and save the project before execution.

Once one gets used to generating `sif` files and running the solver, ElmerGUI includes a double green arrow icon. The double green arrow icon combines all three steps into a single mouse click: Generate and save `sif`, save project, and then run solver.

Postprocessing with ElmerVTK

ElmerVTK is a simple post processor to use, and makes sense for a first look at Elmer simulation results. We will cover Paraview later, which is a more powerful visualization program and, of course, is also a little harder to learn and use.

To view the results we will run ElmerVTK for the visualization.

```
Run
ElmerVTK
```


The initial window for ElmerVTK will pop up, with the surface of the geometry colored in blue, as shown in Figure 1.17. Observe the row of tabs along the top of the window. To hide the surface coloring, left click on the Surfaces tab.

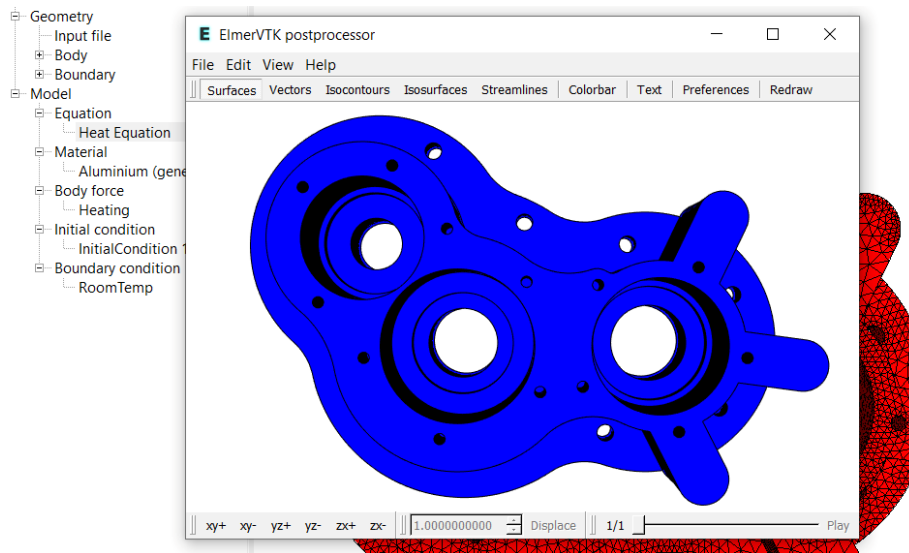


Figure 1.17: The ElmerVTK window, Surfaces

We want to add 3D contouring, so click on the **Iso**surfaces tab. Click in the Contour control Variable box, and select temperature from the drop down box. Continue and click in the Color control Color drop box and again select temperature, as shown in Figure 1.18.

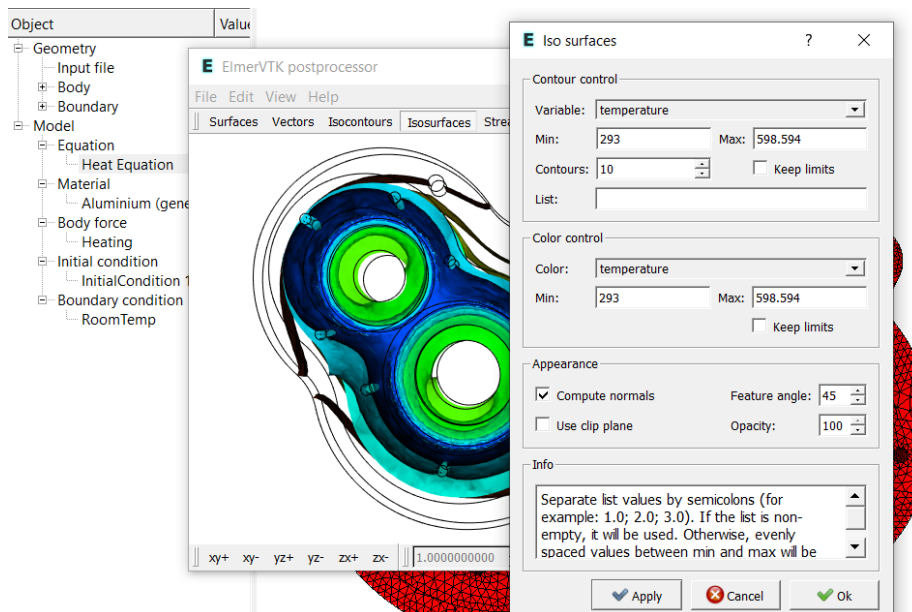


Figure 1.18: The ElmerVTK window, Iso surfaces selection

Click on Apply and Ok, to apply the settings and close the pop up selection window.

The isosurfaces will be displayed, as shown in Figure 1.19. Use the mouse to rotate the part, to see the isosurfaces from different viewpoints.

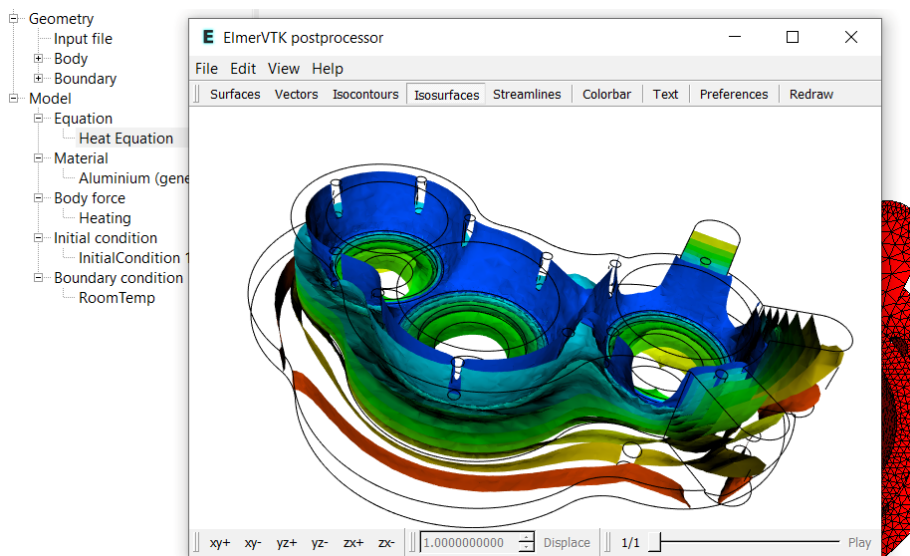


Figure 1.19: The ElmerVTK window, Isosurfaces shown

Let’s add a color bar to the view, so we can see the range of temperatures in the simulation. Click on the Colorbar tab. The Colorbar selection menu will pop up, as shown in Figure 1.20. Click in the Color map drop box, and select Isosurface. Also, click on the radio button for Vertical, to locate the color bar along the left side.

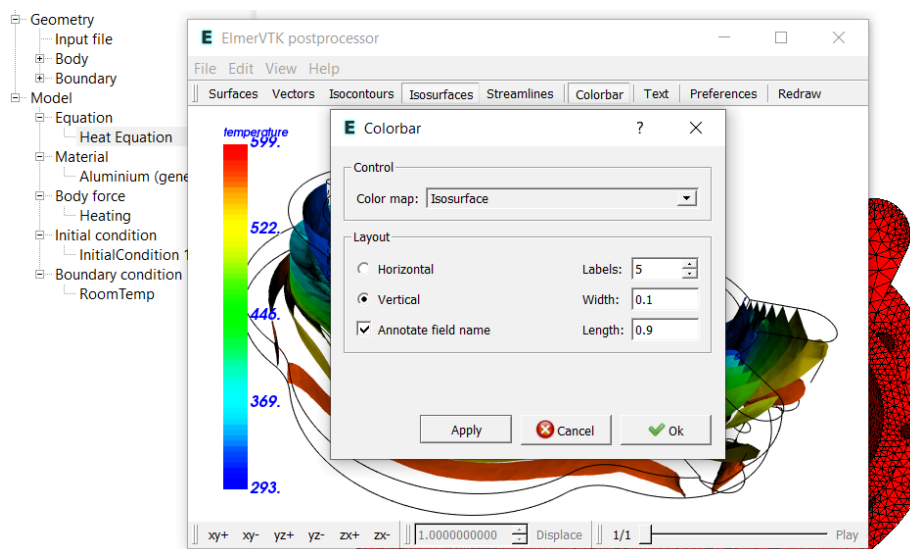


Figure 1.20: The ElmerVTK window, Colorbar selection

Click on Apply and Ok, to apply the settings and close the pop up selection window.

The color bar and the isosurfaces will be displayed, as shown in Figure 1.21.

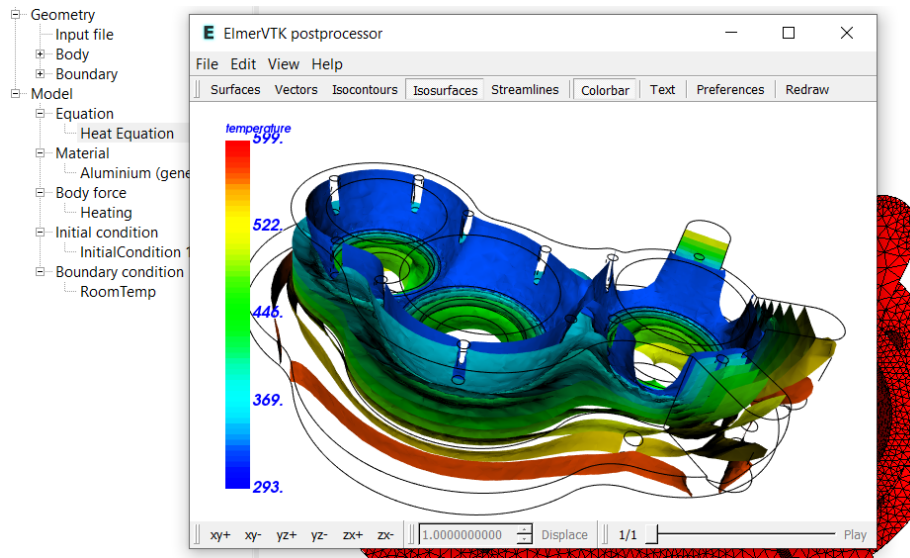


Figure 1.21: The ElmerVTK window, with color bar

To demonstrate how to save a picture with ElmerVTK, click on `File`, `Save picture as`, then select a file name and directory to store the picture, and ElmerVTK will save your simulation results, as shown in Figure 1.22. If the picture is too small or too large for you, just resize the ElmerVTK window and save the picture again.

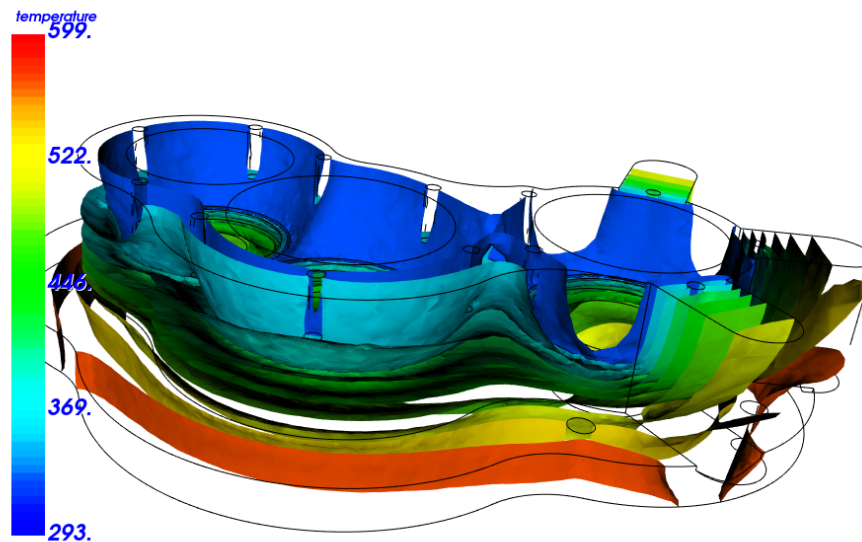


Figure 1.22: The ElmerVTK window, final results

Next we will demonstrate clipping with ElmerVTK, which may be very helpful for 3D simulations. Refer back to the Isosurfaces selection window in Figure 1.18, under the section `Appearances`, click on `Use clip plane`, and then `Apply`. The results are shown in Figure 1.23.

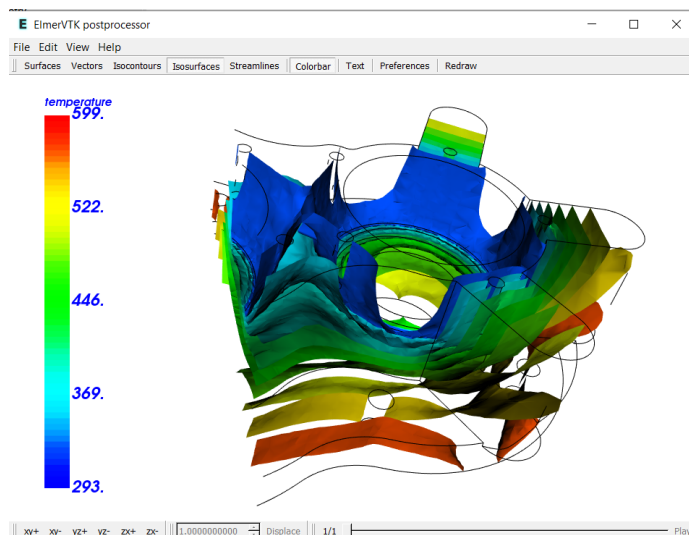


Figure 1.23: The ElmerVTK window, clip plane applied

ElmerGUI supplies a default clip plane setting going in the X direction, through the center of the model. To adjust the clip plane settings, click on the tab `Preferences`, and notice at the bottom are six settings. Adjust the X, Y, or Z coordinates, or adjust the X, Y, or Z normals. Note that one may use a negative normal, such as -1, to clip the model in the opposite direction.

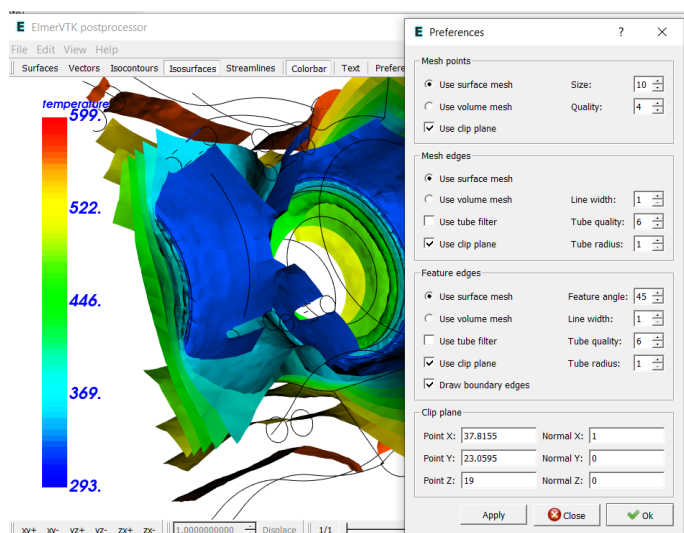


Figure 1.24: The ElmerVTK window, Preferences and clip plane

One can now close ElmerVTK, or spend some time exploring a few more features of ElmerVTK, such as `Vectors`. Also, if one runs a transient study, ElmerVTK can show each of the time steps, using the slider control at the bottom right of the window.

Postprocessing with Paraview

Next we will cover the basics of using Paraview to visualize the simulation results.

Run

Paraview

After a short wait, Paraview will open and it should show our case_t0001.vtu file in the upper left window, as shown in Figure 1.25. First step is to click on the light green Apply button.

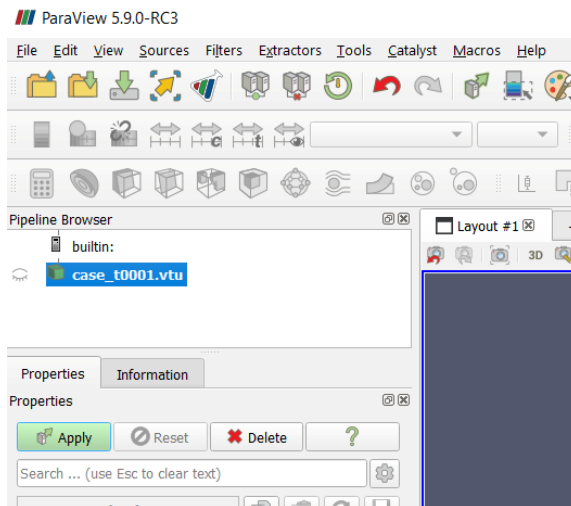


Figure 1.25: The Paraview window, click on Apply

Once we have clicked on Apply, the geometry will appear in the layout port, and will be all grey. We will then tell Paraview which simulation results we wish to display, by clicking on the button under the section header **Coloring** located in the lower left corner, as shown in Figure 1.26. Select temperature in the drop box, as shown in Figure 1.26.

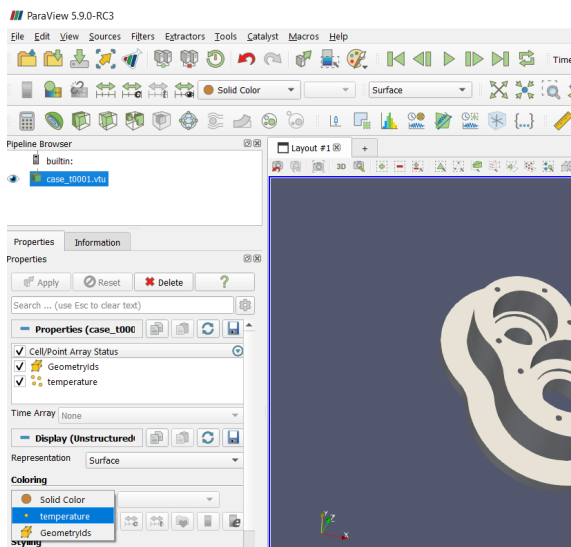


Figure 1.26: The Paraview window, select variable

The temperature of the simulation will be displayed in the layout port, along with a color bar that is automatically added to the layout, as shown in Figure 1.27.

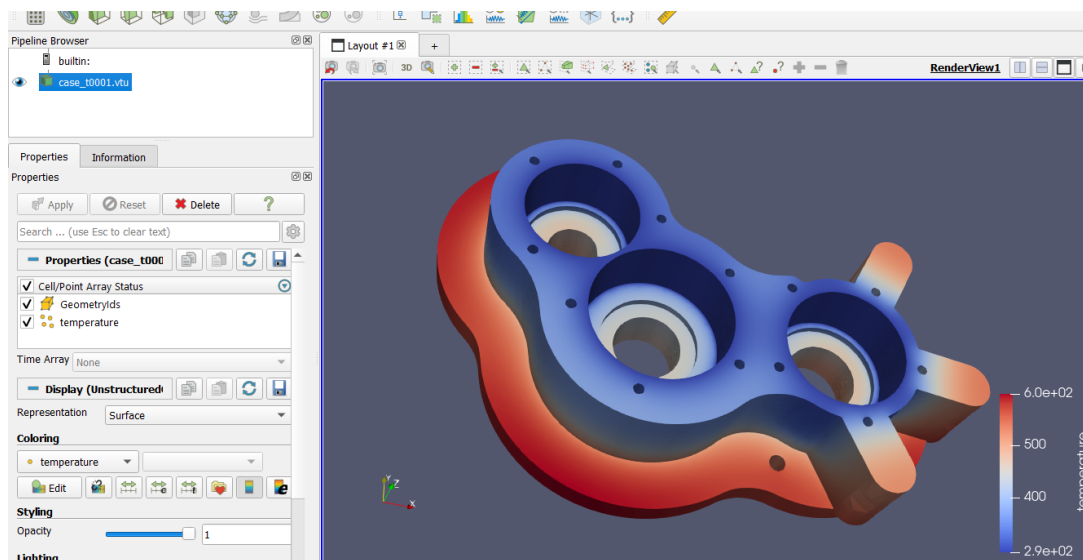


Figure 1.27: The Paraview window, showing results

Lastly, let's save a picture with Paraview. At the top of the window, click on File, then Save Screenshot, as shown in Figure 1.28.

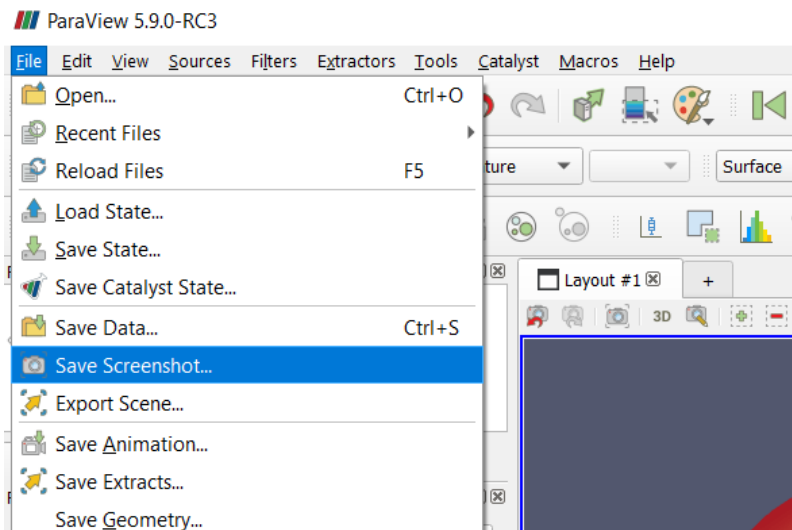


Figure 1.28: The Paraview window, Screenshot

Follow the prompts, such as selecting a file name and directory to store the picture, select the size in pixels of the picture, and Paraview will save your simulation results, as shown in Figure 1.29.

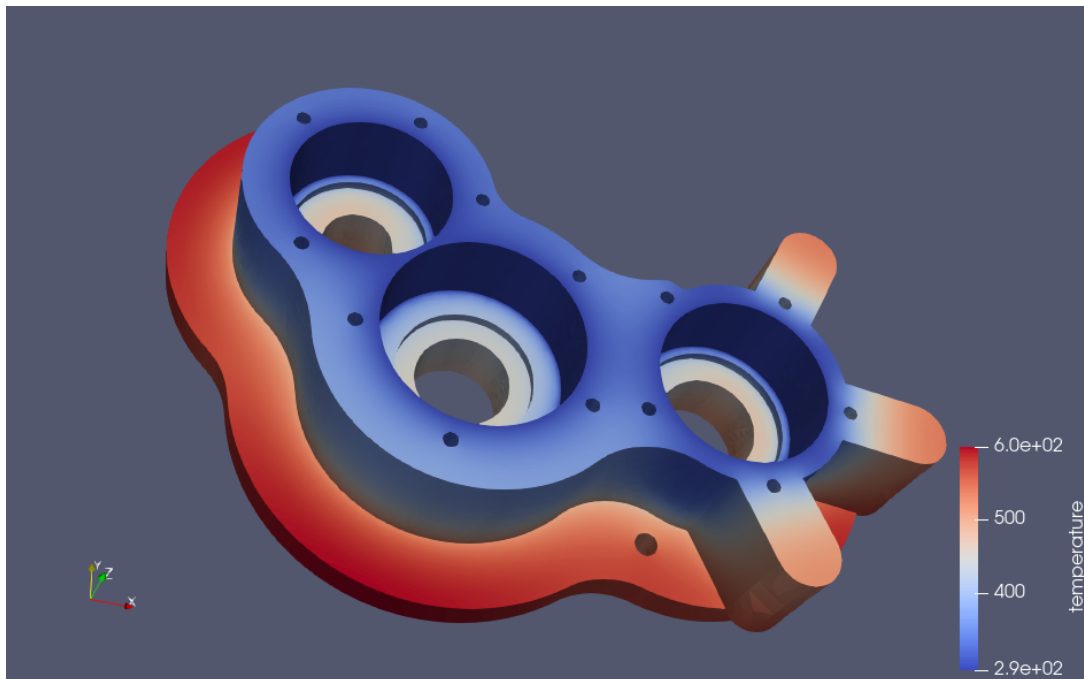


Figure 1.29: The Paraview window, final results

Paraview has many options, as you can see from the many icons in the main window. As one example, we will demonstrate clipping with Paraview, which may be very helpful for 3D simulations. At the top bar, click on *Filters*, *Common*, *Clip*, as shown in Figure 1.30. As always with Paraview, click on the light green *Apply* button to apply the action to the layout.

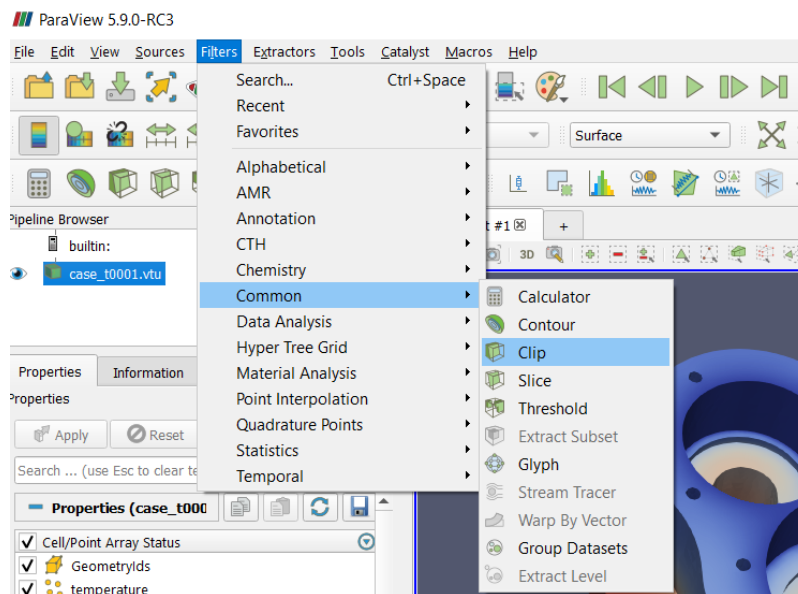


Figure 1.30: The Paraview window, Filter, Common, Clip

One can rotate the model, along with moving the clip plane, or selecting direction with X, Y, or Z normals. These steps will be left to the student to discover.

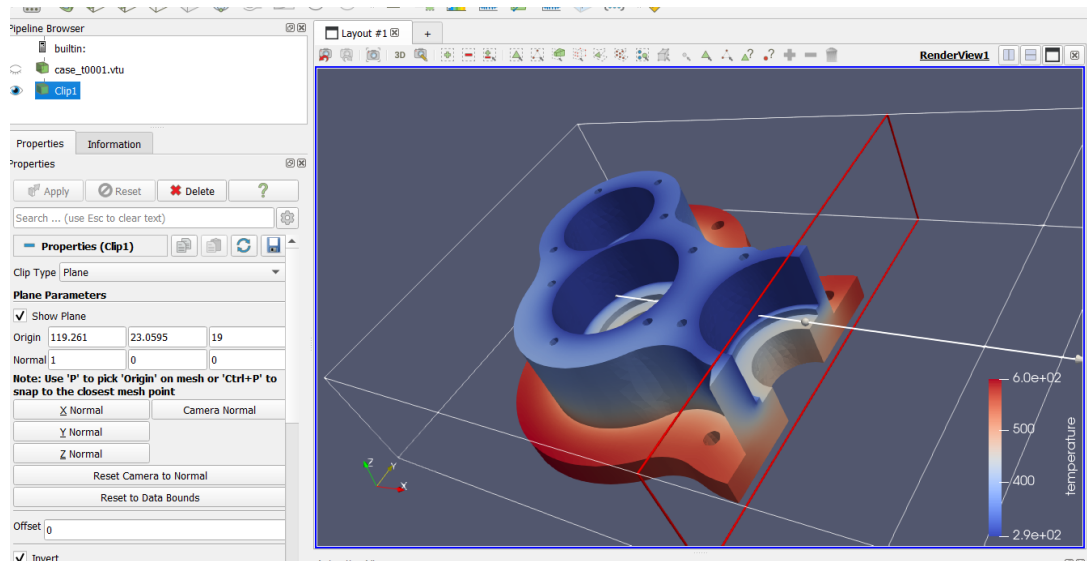


Figure 1.31: The Paraview window, clipped results

This is the end of the introduction to Paraview.

Tutorial 2

Heat equation – 2D – Temperature field of an L-shaped domain

Directory: TemperatureAngleGUI

Solvers: HeatSolve

Tools: ElmerGUI

Dimensions: 2D, Steady-state

Author: Peter Råaback

Problem description

An L-shaped structure (see figure 2.1) is heated by an internal heat source, which magnitude is 1 W/m^3 . The density of the structure is 1 kg/m^3 and the heat conductivity is 1 W/mK . All the boundaries Γ_i are kept on constant temperature of 0 K . The problem is to solve the temperature distribution in the structure. Mathematically the problem to be solved is

$$\begin{cases} -\kappa\Delta T & = \rho f & \text{in } \Omega \\ T & = 0 & \text{on } \Gamma \end{cases} \quad (2.1)$$

where κ is the heat conductivity, T is the temperature and f is the heat source. It is assumed that density and heat conductivity are constants.

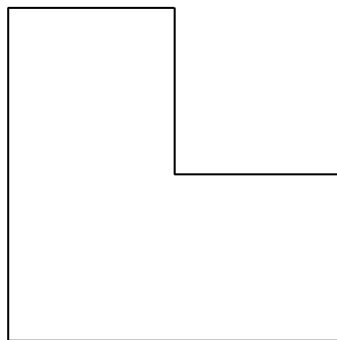


Figure 2.1: L-shaped domain

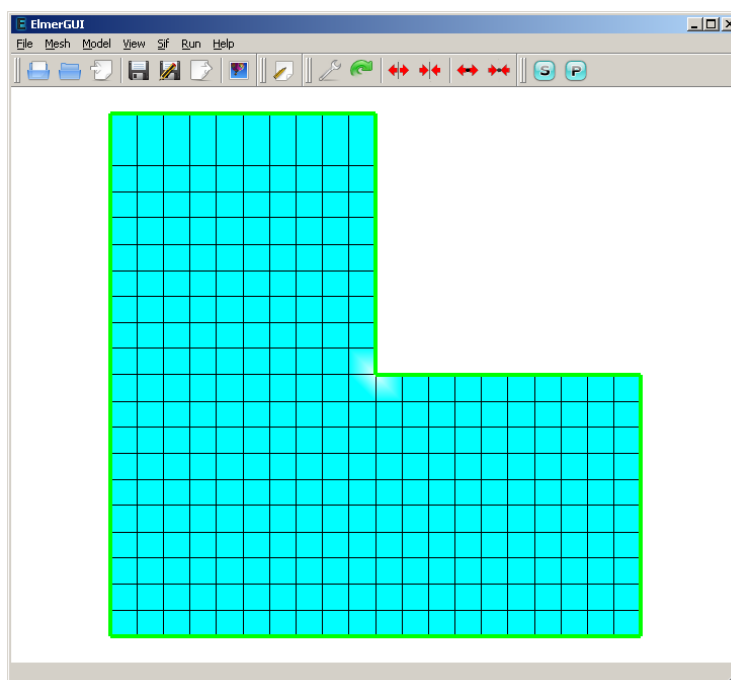


Figure 2.2: The finite element mesh in ElmerGUI

Solution procedure

Start ElmerGUI from command line or by clicking the icon in your desktop. Here we describe the essential steps in the ElmerGUI by writing out the clicking procedure. Tabulation generally means that the selections are done within the window chosen at the higher level.

The mesh is given in ElmerGrid format in file `angle.grd` in the samples directory of ElmerGUI, load this file.

```
File
  Open -> angle.grd
```

You should obtain your mesh and may check in the `Model summary` window that it consists of 341 nodes and 300 bilinear elements. If the mesh was successfully imported your window should look something in figure 2.2.

After we have the mesh we start to go through the `Model` menu from the top to bottom. In the `Setup` we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried out in 2-dimensional Cartesian coordinates and in steady-state. Only one steady-state iteration is needed as the case is linear.

```
Model
  Setup
    Simulation Type = Steady state
    Steady state max. iter = 1
```

Choose `Accept` to close the window.

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have one set only one equation – the heat equation.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and one boundary and therefore its easier to assign the Equation and Material to it directly.

For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1,...) or direct Umfpack solver, for example.

```
Model
Equation
Add
  Name = Heat Equation
  Apply to bodies = 1
  Heat Equation
  Active = on
Apply
OK
```

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such heat conductivity.

```
Model
Material
Add
  Name = Ideal
  Apply to bodies = 1
  General
  Density = 1.0
  Heat Equation
  Heat Conductivity = 1.0
Apply
OK
```

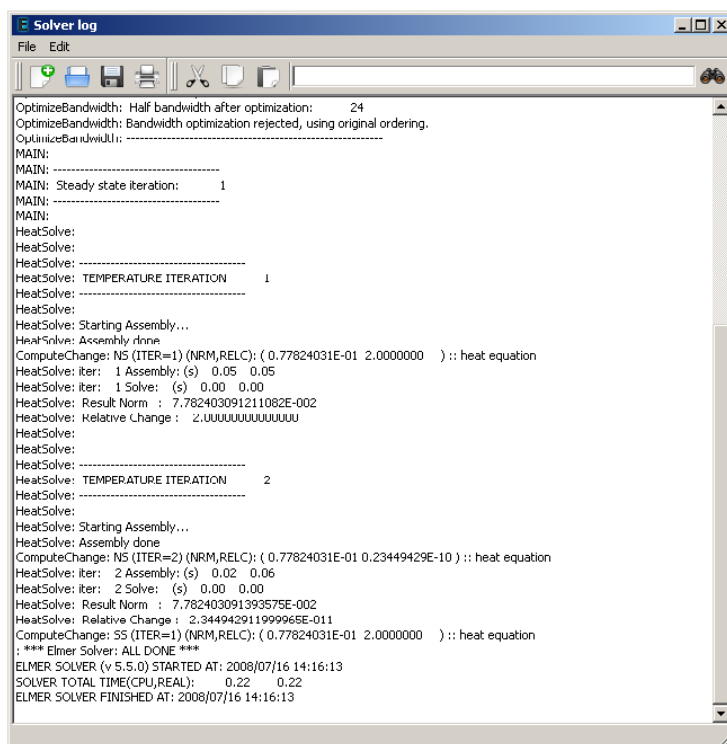
A Body Force represents the right-hand-side of a equation that in this case represents the heat source.

```
Model
Body Force
Add
  Name = Heating
  Heat Source = 1.0
  Apply to bodies = 1
Apply
OK
```

No initial conditions are required in steady state case.
In this case we have only one boundary and set it to zero.

```
Model
BoundaryCondition
Add
  Heat Equation
  Temperature = 0.0
  Name = Zero
  Apply to boundaries = 1
Apply
OK
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.



```

Solver log
File Edit
OptimizeBandwidth: Half bandwidth after optimization: 24
OptimizeBandwidth: Bandwidth optimization rejected, using original ordering.
OptimizeBandwidth: -----
MAIN:
-----
MAIN: Steady state iteration: 1
MAIN: -----
HeatSolve:
-----
HeatSolve:
-----
HeatSolve: TEMPERATURE ITERATION 1
HeatSolve: -----
HeatSolve: Starting Assembly...
HeatSolve: Assembly done
ComputeChange: NS (ITER=1) (NRM,REL): ( 0.77824031E-01 2.0000000 ) :: heat equation
HeatSolve: iter: 1 Assembly: (s) 0.05 0.05
HeatSolve: iter: 1 Solve: (s) 0.00 0.00
HeatSolve: Result Norm : 7.782403091211082E-002
HeatSolve: Relative Change : 2.000000000000
HeatSolve:
-----
HeatSolve: TEMPERATURE ITERATION 2
HeatSolve: -----
HeatSolve: Starting Assembly...
HeatSolve: Assembly done
ComputeChange: NS (ITER=2) (NRM,REL): ( 0.77824031E-01 0.23449429E-10 ) :: heat equation
HeatSolve: iter: 2 Assembly: (s) 0.02 0.06
HeatSolve: iter: 2 Solve: (s) 0.00 0.00
HeatSolve: Result Norm : 7.782403091393575E-002
HeatSolve: Relative Change : 2.344942911999965E-011
ComputeChange: SS (ITER=1) (NRM,REL): ( 0.77824031E-01 2.0000000 ) :: heat equation
*** Elmer Solver: ALL DONE ***
ELMER SOLVER (v 5.5.0) STARTED AT: 2008/07/16 14:16:13
SOLVER TOTAL TIME(CPU,REAL): 0.22 0.22
ELMER SOLVER FINISHED AT: 2008/07/16 14:16:13

```

Figure 2.3: The output log of ElmerSolver when used under ElmerGUI

```

Sif
  Generate
  Edit -> look how your command file came out

```

Before we can execute the solver we should save the files in a directory. In saving the project all the necessary files for restarting the case will be saved to the destination directory.

```

File
  Save Project

```

After we have successfully saved the files we may start the solver

```

Run
  Start solver

```

A convergence view automatically pops up showing relative changes of each iteration. As the case is linear only one iteration was required for the solution and the second one just is needed to check the convergence. The resulting output log is shown in figure 2.3.

Note: if you face problems in the solution phase and need to edit the setting, always remember to save the project before execution.

To view the results we use Paraview.

```

Run
  Start Paraview

```

Picture 29.2 shows the the surface mesh colored with temperature (the pictures were generated by the obsolete ElmerPost software).

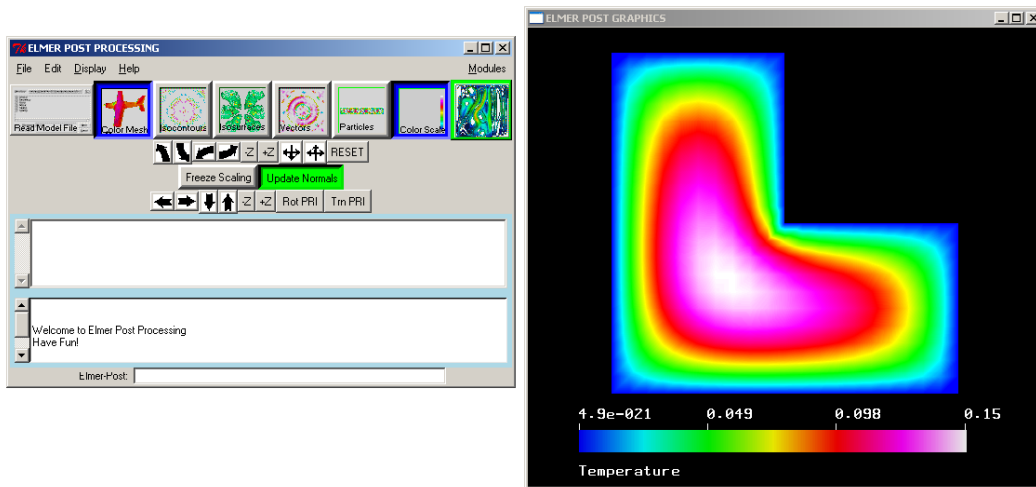


Figure 2.4: The temperature distribution of the L-shaped domain.

Results

As a reference result the maximum temperature in the structure is given. For a comparison the same problem was solved six times with different element sizes using the `-relh` flag of ElmerGrid format located in the Mesh menu under Configure. After choosing Remesh and saving the mesh the solver may be recalled with the modified mesh.

The maximum temperature obtained by using different meshes is recorded in Table 2.1. From the results one can see that the result converges. With a denser mesh the result is more accurate, but solving the problem takes more calculation time. For reference, the CPU-time used in each case is also shown in the table (using Lenovo 60 laptop).

Table 2.1: Results with different element sizes

Model mesh h [m]	Elements	$\max(T)$ [K]	cpu time [s]
0.2	75	0.1452	0.14
0.1	300	0.1477	0.19
0.05	1200	0.1489	0.41
0.025	5120	0.14925	1.22
0.01	30800	0.14937	7.77

Tutorial 3

Heat Equation –1D – Temperature of an idealized geological intrusion

Directory: HeatIntrusion1D

Solvers: HeatSolve

Tools: ElmerGUI

Dimensions: 2D, Transient

Author: Peter Råback, Thomas Zwinger

Problem description

This is an extreme simplification of the intrusion process in related to geological processes. The case is effectively 1D but it is treated as 2D for generality of the approach.

We study temperature distribution from -8 km to -4 km. It is assumed that before the intrusion there is a linear temperature distribution from 320 to 160 C. At $t = 0$ an intrusion of 900 C replaces the temperature from -6.5 km to -5.5 km. The initial temperature distribution may therefore be presented by a piecewise linear function passing through the points depicted in table 3.1.

Table 3.1: Piecewise linear function defining the initial temperature

depth [m]	T (C)
-8000	320.0
-6500	260.0
-6499	900.0
-5501	900.0
-5500	220.0
-4000	160.0

The material parameters for the rock are assumed to be those of ideal granite. We take $k = 2.5$ W/mK, $C_p = 1250$ J/kgK, and $\rho = 2800$ kg/m³.

As boundary conditions we use the initial temperature at the upper and lower end. Hence the problem depends only on the depth direction.

Solution procedure

Start ElmerGUI from command line or by clicking the icon in your desktop. Here we describe the essential steps in the ElmerGUI by writing out the clicking procedure. Indentation generally means that the selections are done within the window chosen at the higher level.

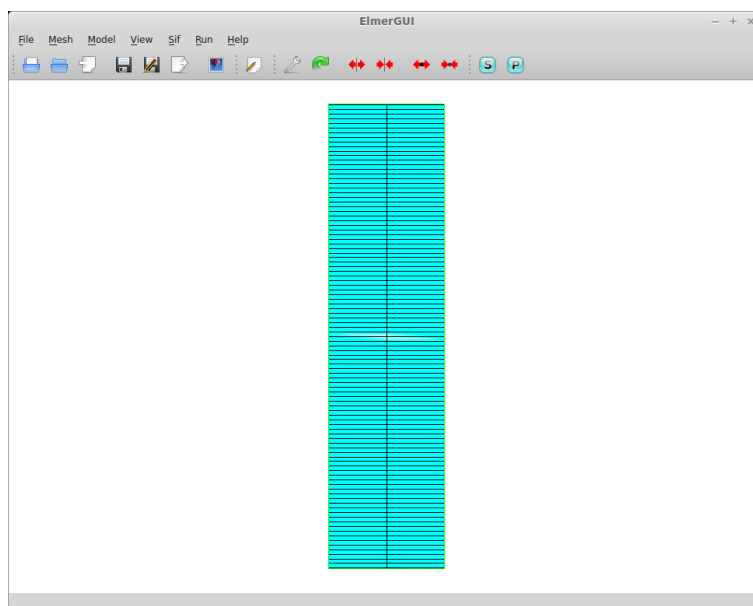


Figure 3.1: The geometry of the mesh in ElmerGUI

The mesh is given in ElmerGrid format in file `geoslab.grd` in the samples directory of ElmerGUI, load this file.

File

```
Open -> geoslab.grd
```

You should obtain your mesh and may check in the `Model summary` window that it consists of 303 nodes and 200 quadrilateral surface elements. The mesh is very coarse in the x -direction intentionally since we are looking just for 1D solution. If the mesh was successfully imported your window should look something in figure 3.1.

After we have the mesh we start to go through the `Model` menu from the top to bottom. In the `Setup` we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried out in 2-dimensional Cartesian coordinates and in transient. We will use 1000 time steps each of size 10 years. For time stepping we will use 2nd order scheme.

Model

Setup

```
Simulation Type = Steady transient
Timestepping Method = BDF
BDF Order = 2
Timestep Intervals = 10000
Timestep Sizes = $10*365*24*3600
Output Intervals = 2
```

Choose `Apply` to accept the values and close the window.

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have only one equation – the heat equation.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and one boundary and therefore its easier to assign the Equation and Material to the body directly.

For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1, ...) or direct Umfpack solver, for example.

```

Model
  Equation
    Add
      Name = Heat Equation
      Apply to bodies = 1
      Heat Equation
        Active = on
    Update
  OK

```

The Material section includes all the material parameters. They are divided to generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such heat conductivity.

```

Model
  Material
    Add
      Name = Granite
      Apply to bodies = 1
      General
        Density = 2700.0
        Heat Capacity = 1250.0
      Heat Equation
        Heat Conductivity = 2.5
    Update
  OK

```

We will need an initial condition for the case.

```

Model
  Initial Condition
    Add
      Name = Initial State
      Apply to bodies = 1
      Heat Equation
        Temperature -> click in the box, press enter to open a text box and write the
    Apply
  OK

```

Now the expression to be written uses an internal linear interpolation feature of Elmer. The interpolation expression will be entered into the 'Variables' box named 'Temperature'. Click in the 'Temperature' box and press enter. A pop up window that will accept text input will open. Enter the below text, and click on 'close' to accept. Note, do not try to enter the expression into the 'Free text input' box right below 'Temperature', it will not work.

```

Variable coordinate 2
  Real
    -8000 320.0
    -6500 260.0
    -6499 500.0
    -5501 500.0
    -5500 220.0
    -4000 160.0
  End

```

In this case we only need boundary conditions for the upper and lower boundary. First we create the boundary conditions


```

Model
  BoundaryCondition
    Add
      Heat Equation
        Temperature = 320.0
        Name = Down
      OK
    Add
      Heat Equation
        Temperature = 160.0
        Name = Up
      OK

```

Then we set the boundary properties

```

Model
  Set boundary properties

```

Choose the defined group of three boundaries by clicking with the mouse and apply the condition for the lower boundary.

```

Boundary condition
  Down

```

and similarly for the upper boundary.

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```

Sif
  Generate
  Edit... -> look how your command file came out

```

Before we can execute the solver we should save the files in a directory. In saving the project all the necessary files for restarting the case will be saved to the destination directory.

```

File
  Save Project

```

After we have successfully saved the files we may start the solver

```

Run
  Start solver

```

A convergence view automatically pops up showing relative changes of each iteration. As the case is linear only one iteration was required for the solution and the second one just is needed to check the convergence. The resulting output log is shown in figure 3.2.

Note: if you face problems in the solution phase and need to edit the setting, always remember to save the project before execution.

To view the results we use Paraview.

```

Run
  Start ParaView

```

If your configuration is OK a Paraview window should pop up. Choose temperature for the surface to be plotted. As we have a time series we may run the sequence by pressing the play icon.

You may also choose the Plot Over Line filter to see the data plotted over a line for better numerical inspection.

The final maximum temperature of the analysis is around 600 C. Whether this is close to correct could be studied by increasing further the time and space resolution.

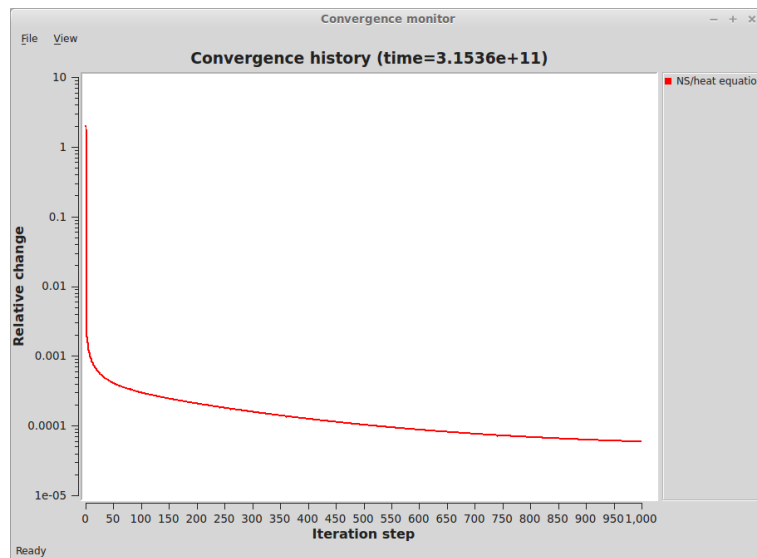


Figure 3.2: The output log of ElmerSolver when used under ElmerGUI

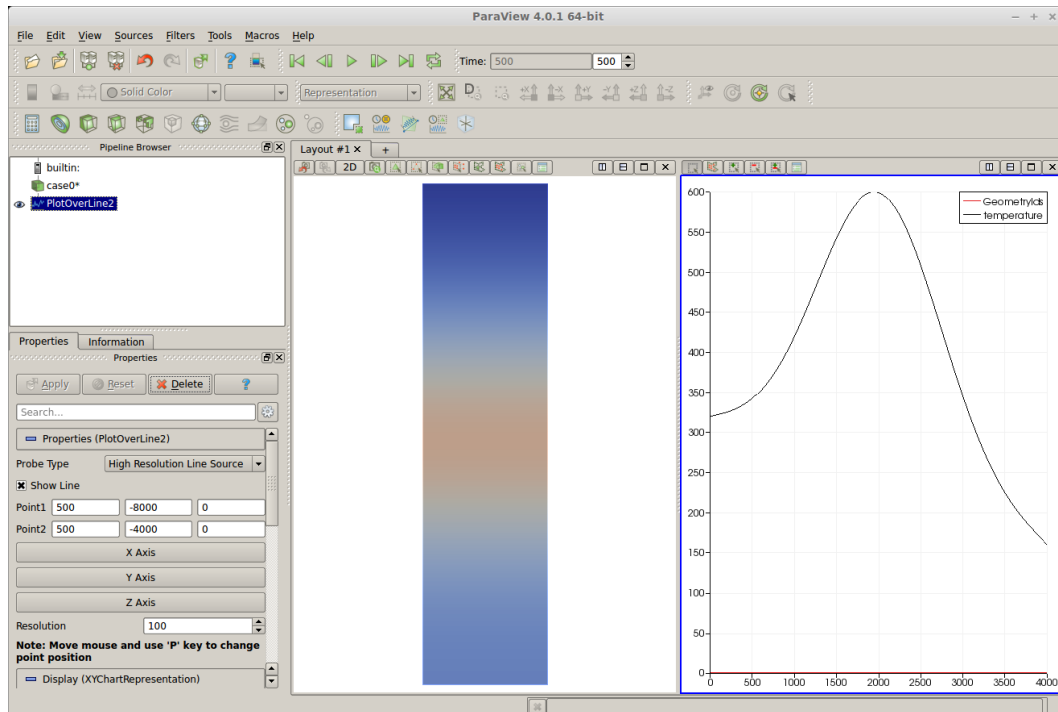


Figure 3.3: The data visualized in Paraview as a surface and line plot

3.0.1 Variation accounting for latent heat release

For intrusion processes the latent heat often plays an important role. For that the internal phase change model can be used. It requires that all the internal energy is expressed using specific enthalpy rather than heat capacity and latent heat.

We now eliminate the heat capacity of the original case (1250 J/kgK) and create a specific enthalpy that includes heat capacity of 1000 J/kgK and latent heat release of 200 kJ/kg release between interval 700–800 C.

The expression for Specific Enthalpy accounting for these two now yields

```
Variable Temperature
Real
  0.0  0.0
  700  7.0e5
  800  9.0e5
  900  10.0e5
End
```

For the phase change model we select

```
Model
Equation
  Heat Equation
  Phase Change Model = spatial 2
Update
OK
```

With these changes the maximum temperature at the end of simulation cycle becomes around 624 C.

Tutorial 4

Heat Equation – 2D – Axi Symmetric Steady State Radiation

Directory: Radiation

Solvers: HeatSolve

Tools: ElmerGUI

Dimensions: 2D, Axi-Symmetric

Author: Juha Ruokolainen, Rich Bayless

Case definition

At high temperature the radiation heat transfer between walls is often the dominating heat transfer mechanism. In this tutorial we examine how radiation heat transfer between concentric cylinders is modelled.

The problem is a pure heat transfer problem that may be solved with `HeatSolve`. The view and Gebhart factors associated with the radiation are solved as a first step in solving the equations. Thereafter the non-linear heat equation is solved until convergence is reached.

Solution procedure

The mesh is given in ElmerGrid format in file `radiation.grd`, load this file.

File

```
Open -> radiation.grd
```

You should obtain your mesh and may check `Model Summary...` that it consists of 1231 surface elements. Your geometry and mesh should look something like as shown in figure 31.1

After we have the mesh we start to go through the `Model` menu from the top to bottom. In the `Setup` we choose things related to the whole simulation such as file names, time stepping, constants etc.

The simulation is carried out in `Axi Symmetric` coordinates. The only constant required is the Stefan-Boltzmann constant that gives the relationship between temperature and radiation power, and this constant is predefined in the `Setup` menu.

Model

Setup

```
Coordinate System = Axi Symmetric
```

```
Simulation Type = Steady State
```

```
Steady State Max Iterations = 1
```

```
Output Intervals = 1
```

Apply

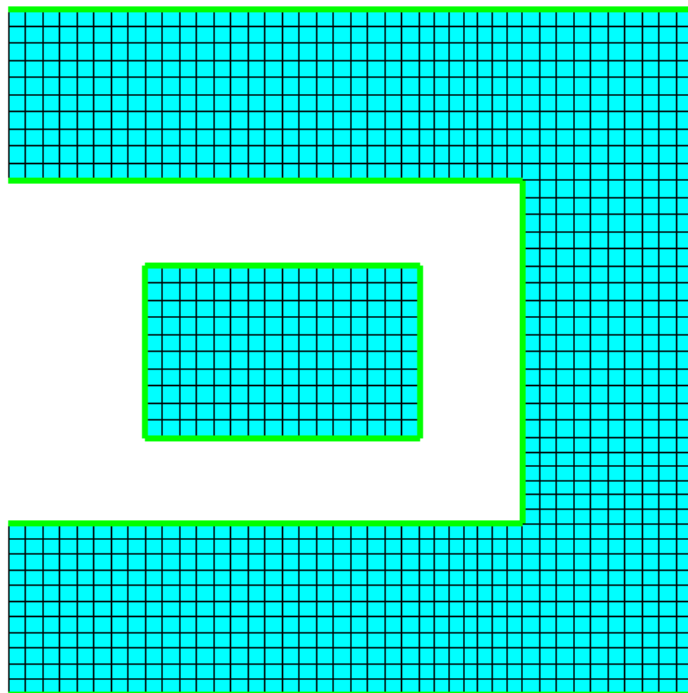


Figure 4.1: Geometry and mesh

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have the Heat equation.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just two bodies and therefore its easier to assign the Equation and Material to it directly.

For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1,..) or direct Umfpack solver, for example.

Model

Equation

Name = Radiation

Apply to Bodies = 1 2

Heat Equation

Active = on

Edit Solver Settings

Steady State

Convergence Tolerance = 1.0e-8

Non-linear system

Convergence Tolerance = 1.0e-8

Max Iterations = 50

Relaxation Factor = 0.7

Newton After Iterations = 1

Newton After Tolerance = 1.0e-4

Linear system

Convergence Tolerance = 1.0e-12

Preconditioning = ILU1

Add

OK

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such as conductivities and viscosity.

The material properties differ only in heat conductivity. Heat capacity is not actually needed since the case is not transient. The inner body has ten times higher conductivity than the outer body.

Model

```
Material
  Name = Inner
  Apply to Bodies = 1
  General
    Density = 1.0
    Heat capacity = 1.0
  Heat Equation
    Heat Conductivity = 10.0
  Add
  New

  Name = Outer
  Apply to Bodies = 2
  General
    Density = 1.0
    Heat capacity = 1.0
  Heat Equation
    Heat Conductivity = 1.0
  Add
  OK
```

A Body Force represents the right-hand-side of a equation. The body force is the heating power in units W/kg.

Model

```
Body Force
  Name = Power
  Apply to Bodies = 1
  Heat Equation
    Volume Heat Source = 10000
  Add
  OK
```

Initial conditions are needed in this case. We choose a constant Temperature field of 250C.

Model

```
Initial Condition
  Name = Initial
  Apply to Bodies = 1 2
  Heat Equation
    Temperature = 250.0
  Add
  OK
```

Only one boundary condition may be applied to each boundary and therefore all the different physical BCs for a boundary should be grouped together.

The radiation boundary conditions are set for two different boundaries. The first one is for the internal heated object and the second one for the outer insulating body. The normal direction of the surfaces is important since a wrong direction may result to a badly set problem for the view factor computation. Internal and external surfaces are very different. The normal direction may be switched with the input box `Radiation Target Body`. A good sign of a properly set case is that the view factors add up to about one.

The third boundary condition is the Dirichlet condition for the external boundary. Dirichlet conditions boost up the convergence even though the heat equation is basically well defined also with external radiation conditions.

Model

```
BoundaryCondition
  Name = Inner
  Apply to Bodies = 1
  Heat Equation
    Radiation Settings = Diffuse Gray
    Emissivity = 0.6
    Radiation Target Body = -1
  Add
  New

  Name = Outer
  Apply to Bodies = 2
  Heat Equation
    Radiation Settings = Diffuse Gray
    Emissivity = 0.1
    Radiation Target Body = -1
  Add
  New

  Name = Exterior
  Apply to Bodies = 3
  Heat Equation
    Temperature = 100.0
  Add
  OK
```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

Model

```
Set boundary properties
  Choose Inner -> set boundary condition Inner
  Choose Outer -> set boundary condition Outer
  Choose Exterior -> set boundary condition Exterior
Update
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

Sif

```
Generate
Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The ElmerGUI project includes all the files needed to restart the case.

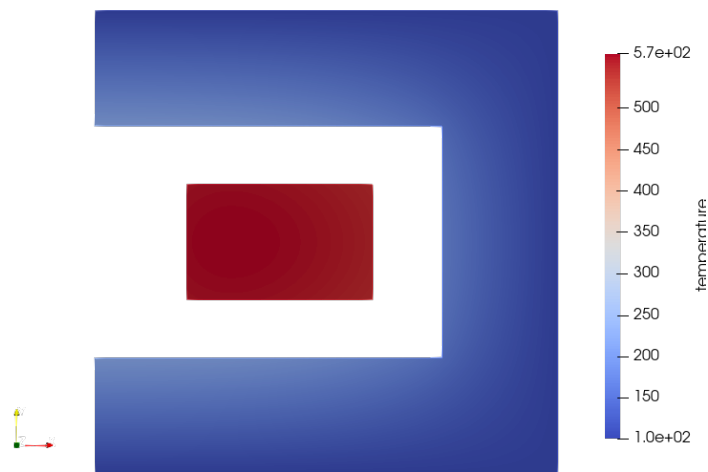


Figure 4.2: Temperature distribution in the radiation heat transfer problem

File

Save Project

After we have successfully saved the files we may start the solver

Run

Start solver

A convergence view automatically pops up showing relative changes of each iteration.

When there are some results to view we may start the postprocessor also

Run

Start ParaView

Results

With the given computational mesh the problem is solved in a few seconds. With 1,231 second order 9-noded rectangular elements the maximum temperature is 565.7 K. The corresponding results are shown in Fig. 4.2.

Tutorial 5

Heat Equation – 2D – Active and Passive elements

Directory: PassiveElements
Solvers: HeatSolve
Tools: ElmerGUI
Dimensions: 2D, Transient
Author: original author not known

Case definition

This tutorial shows an example of using passive elements in Elmer. This feature allows the activation and deactivation of parts of the geometry during the simulation. This tutorial uses the heat equation solver to demonstrate this capability. Use with other solvers is analogous.

The geometry of the problem consists of two parts. The lower edge of the lower part is held at constant temperature of 1 degrees. The upper body is heated with a constant heating power. Between time steps 5 and 6 the two bodies are connected by two heat conductors, and the heat is conducted from the higher body to the lower one. The goal of the simulation is to model the temperature distribution in the system over time.

The problem is a pure heat transfer problem that may be solved with `HeatSolve`.

The geometry for this tutorial looks like as shown in figure 31.1.

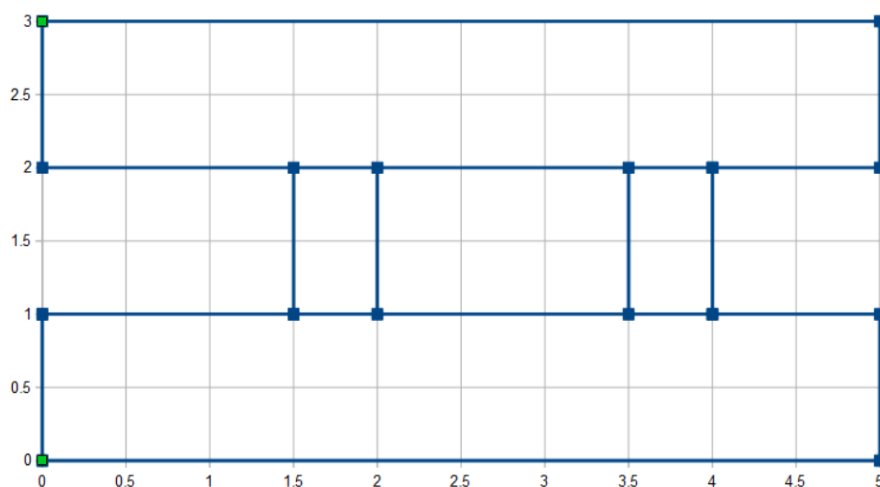


Figure 5.1: Geometry

ElmerGUI Equation Menu

We will be using the Heat Solver, which is one of the default, pre-loaded GUI definitions, so it does not need to be manually activated. For reference, the `heatequation.xml` definition file is, by default, located in Linux in:

```
$ELMER_HOME/share/ElmerGUI/edf
```

and in Windows, located in:

```
C:/Program Files/Elmer 9.0-Release/share/ElmerGUI/edf
```

Solution procedure

The mesh is given in ElmerGrid format in file `tmesh.grd`, load this file.

File

```
Open -> tmesh.grd
```

You should obtain your mesh and may check `Model Summary...` that it consists of 2018 surface elements. Your mesh should look like as shown in figure 31.5

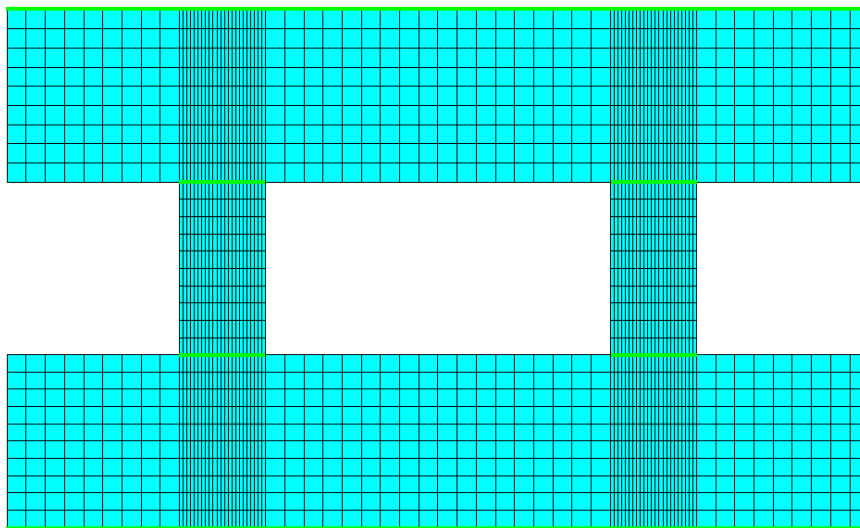


Figure 5.2: Mesh

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc.

The simulation is carried out in 2-dimensional Cartesian coordinates.

2nd order bdf time stepping method is selected with 15 steps and with step size of 1 seconds.

The heat equation is solved with an iterative method. The system is linear, thus multiple non-linear iterations are not needed.

Model

```
Setup
```

```

Simulation Type = Transient
Steady state max. iter = 1
Time Step Intervals = 15
Time Step Sizes = 1
Apply

```

In the equation section we choose the relevant equations and parameters related to their solution.

In this case we'll have one equation (named "Heat") which consists of the heat equation.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just three bodies and therefore its easier to assign the Equation and Material to it directly.

For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1,..) or direct Umfpack solver, for example.

```

Model
Equation
Name = Heat
Apply to Bodies = 1 2 3
Heat Equation
Active = on
Add
OK

```

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such as conductivities and viscosity.

The material properties of the system are artificial. The following three properties are needed for each material.

```

Model
Material
Apply to Bodies = 1 3
General
Density = 1
Heat Capacity = 1
Heat Equation
Heat Conductivity = 1
Add
OK

```

```

Material
Apply to Bodies = 2
General
Density = 1
Heat Capacity = 10
Heat Equation
Heat Conductivity = 1
Add
OK

```

A Body Force represents the right-hand-side of a equation. It is generally not a required field for a body.

There are three bodies with the same equation but different material properties. Body 3 is heated by a constant body force. Body 2 forms the connecting parts of the system. An initial condition as well as a body force is defined for this body. The body force contains the initial deactivation, and later activation, of the connecting part. Note that this part is included in the geometry all the time, but the passive command is used to define when they are included into the simulation.

Now the passive condition for the connecting part is defined. When the parameter `Temperature Passive` has a value larger than zero, the current element is excluded from the solution, otherwise it is included as a normal element. The parameter may depend on variables, coordinates or time. Here it is defined to depend on time using a tabular format.

The section 'Free text input' is used, because the option to define a passive element has not been added to the ElmerGUI menu file 'heatequation.xml'. For any solver, one can always add additional commands by using the Free text input box.

Model

```
Body Force
  Name = Heat Source
  Apply to Bodies = 3
  Heat Equation
    Volume Sources
      Heat source = 10
  Add
  OK
```

```
Body Force
  Name = Passive
  Apply to Bodies = 2
  Heat Equation
    Free text input
      Temperature Passive = Variable time
      Real
        0.0    1.0
        5.0    1.0
        5.2   -1.0
        8.0   -1.0
    End
  Add
  OK
```

Initial conditions should be given to transient cases, and probably are not needed for steady state solutions.

In this case we choose a constant `Temperature` for the passive element. The initial condition for the initially passive elements is taken to be 5 degrees; a temperature halfway between the colder part and the hotter part of the system.

Model

```
Initial Condition
  Name = Initial Guess
  Apply to Bodies = 2
  Heat Equation
    Temperature = 5.0
  Add
  OK
```

Only one boundary condition may be applied to each boundary and therefore all the different physical BCs for a boundary should be grouped together.

The boundary conditions are simple. The lower boundary of the lower body is held at 1 degrees and the upper boundary of the upper body at 10 degrees. The side walls are assumed to be adiabatic.

Model

```
Boundary Condition
  Name = Bottom
```

```
Heat Equation
  Temperature = 1.0
Add
New
```

```
Boundary Condition
  Name = Top
  Heat Equation
    Temperature = 10.0
  Add
OK
```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking on each boundary with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

```
Model
  Set boundary properties
    Choose Bottom -> set boundary condition Bottom
    Choose Top -> set boundary condition Top
  OK
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
  Generate
  Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The ElmerGUI project includes all the files needed to restart the case.

```
File
  Save Project
```

After we have successfully saved the files we may start the solver.

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. When there are some results to view we may start the postprocessor also.

```
Run
  Start ParaView
```

Results

Due to the number of the time steps the simulation may take around 15 seconds.

You may inspect the results with Paraview or with ElmerVTK.

In Figure 22.4 the obtained temperature distribution is presented, where the left image shows the passive condition, and the right image shows the active condition.

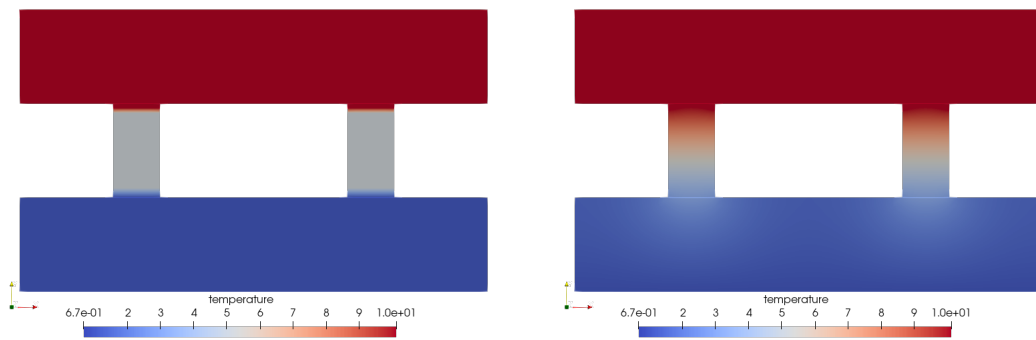


Figure 5.3: Temperature distribution at 4 and 6 seconds

Tutorial 6

Linear elasticity equation – 2D – Loaded elastic beam

Directory: ElasticBeam2D

Solvers: StressSolve

Tools: ElmerGUI

Dimensions: 2D, Steady-state

Author: Peter Råback

Case definition

A homogenous, elastic beam (Ω) is rigidly supported on one end (boundary Γ_4). On boundary Γ_3 the beam is subjected to a load $q(x)$, which grows linearly from zero to q_0 (see figure 6.1). The length of the beam is 1 m and the thickness 0.1 m. Material properties of the beam are those of iron (in the database) Poisson ratio 0.29 and Young's modulus $193 \cdot 10^9 \text{N/m}^2$. Problem is to solve the displacement of the beam.

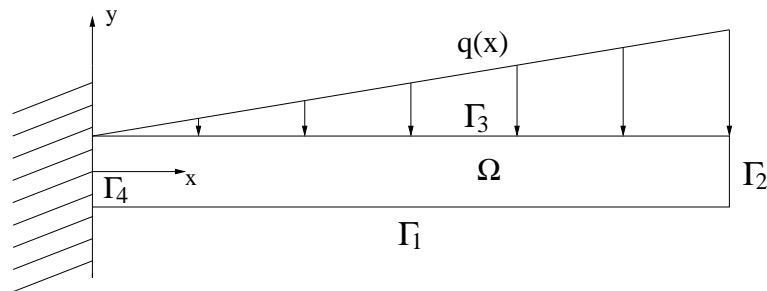


Figure 6.1: Beam and loading.

Problem is solved according to linear elasticity theory. Mathematically the problem to be solved is

$$\left\{ \begin{array}{ll} -div\sigma = 0 & \text{in } \Omega \\ \sigma = \lambda tr[\varepsilon(u)]I + 2\mu\varepsilon(u) & \text{in } \Omega \\ u = 0 & \text{on } \Gamma_4 \\ \sigma n = 0 & \text{on } \Gamma_1 \cup \Gamma_2 \\ \sigma n = -q & \text{on } \Gamma_3 \end{array} \right. \quad (6.1)$$

where λ and μ are the Lamé constants (which can be expressed in terms of the Poisson ratio and Young's modulus), ε is the linearised strain tensor, u is the displacement vector, q is the given surface traction and n is the outward unit normal to the boundary.

Solution procedure

The mesh is given in ElmerGrid format in file `beam.grd`, load this file.

File

```
Open -> beam.grd
```

You should obtain your mesh, `Model, Summary...` and may check that it consists of 1000 biquadratic elements.

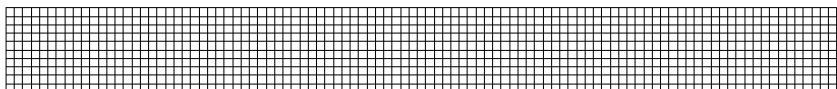


Figure 6.2: The mesh used in the computations

After we have the mesh we start to go through the `Model` menu from the top to bottom. In the `Setup` we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried in steady-state in 2-dimensional Cartesian coordinates.

Model

Setup

```
Simulation Type = Steady state
Steady state max. iter = 1
```

In the `Equation` section we choose the relevant equations which in this case only includes the `Linear elasticity` equation which solves the problem according to linear elastic theory. In this case we assume that the beam is thin in the z -direction and hence assume plane stresses. When defining `Equations` and `Materials` it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and therefore its easier to assign the `Equation` and `Material` to it directly. For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (`ILU1,...`) or direct `Umfpack` solver, for example.

Model

Equation

```
Name = Elasticity
Apply to Bodies = 1
Linear elasticity
  Active = on
  Plane Stress = True
Add
OK
```

The `Material` section includes all the material parameters. They are divided to generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such as conductivities and viscosity.

Here we choose the generic iron from the material library. You may click through the material parameters of the various solvers to ensure that the properties are indeed as they should be. Any consistent set of units may be used in Elmer. The natural choice is of course to perform the computations in SI units.

Model

Material


```

Material library
  Iron (generic)
Apply to Bodies = 1
Add
OK

```

There are no body forces and convergence should be easily obtained with the default initial condition i.e. zero for all fields.

The first boundary condition fixes the beam rigidly at the wall. The second boundary condition may seem confusing. Basically it represents the form using a linear interpolation between two points. At $x = 0$ the force $f_y = 0$ and at $x = 1$ the force $f_y = -1.0e7$, respectively. The semicolon is an alternative separator to line break. To fill in the second boundary condition press Enter in the Force 2 checkbox.

Model

```

BoundaryCondition
  Name = Wall
  Linear elasticity
    Displacement 1 = 0.0
    Displacement 2 = 0.0
  Add
  New

  Name = Top
  Linear elasticity
    Force 2 = Variable Coordinate 1; Real; 0 0; 1 -1.0e7; End
  Add

```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

Model

```

Set boundary properties
  Choose left-hand-side -> set boundary condition Wall
  Choose top -> set boundary condition Top

```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

Sif

```

Generate
Edit -> look how your command file came out

```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case.

File

```

Save Project

```

After we have successfully saved the files we may start the solver

Run

```

Start solver

```

A convergence view automatically pops up showing relative changes of each iteration. Two iterations are performed, the second one only to ensure convergence at the nonlinear level. The norm of the finished computation should be around 0.01808.

When there are some results to view we may start the postprocessor also

Run

```

Start ParaView

```

Results

As a result the absolute value of maximum displacement is given. The displacements calculated with different load values q_0 are tabulated in table 6.1. Note that the absolute value of the displacement varies linearly with respect to the load since the model is linear.

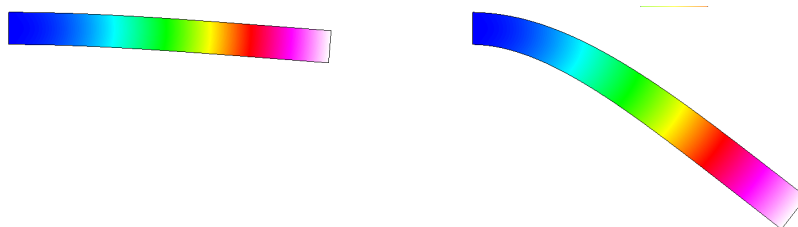


Figure 6.3: The displacement of an elastic beam with two different loads using a linear model

Table 6.1: Displacements with different load values

q_0 [N/m ²]	$\max u $ [m]
-1.0e7	0.05755
-1.0e8	0.5755
-1.0e9	5.755

If you look at the results you can see that the displacement values become relatively large. The linear theory is valid only to small displacements. From Fig 6.3 you can also notice that the beam does not maintain its original form. This means that the linear elasticity theory can not take into consideration all the necessary phenomena that are related to the problem, any more. To be able to solve the problem we must use general elasticity theory.

Extra task: transient loading

If you have time you may try to solve the case using transient loading. You may try with the following settings.

```
Model
  Setup
    Simulation Type = Transient
    Time Stepping Method = bdf
    BDF Order = 2
    Time Step Intervals = 200
    Time Step Sizes = 1.0e-4
```

Having a too large time step will not accurately describe the transient behaviour while having a too small time step consumes unnecessary resources. There is only numerical damping in the system and hence it will oscillate for some time.

Tutorial 7

Linear elasticity equation – 3D – Loaded elastic beam

Directory: ElasticBeam3D

Solvers: StressSolve

Tools: ElmerGUI

Dimensions: 3D, Steady-state

Author: Peter Råback

Case definition

Assume a homogenous, elastic beam being rigidly supported on one end. On the other end it is subjected with a load of 2000 N resulting from an attached object in the gravitational field. The gravity affects also the beam itself. The length of the beam is 1 m and the thickness is 0.05 m, and the width 0.1 m. Material properties of the beam are those of dry pine timber: Poisson ratio 0.37, Young's modulus $10 \cdot 10^9 \text{N/m}^2$, and density 550kg/m^3 . The problem is to solve the displacement and stress field of the beam. Here the `StressSolve` routine based on the linear theory of elasticity is applied.

Solution procedure

The mesh is given in ElmerGrid format in file `beam3d.grd`, load this file.

File

Open -> `beam3d.grd`

You should obtain your mesh and may check that it consists of 6073 nodes and of 1200 quadratic hexahedral elements. The second order elements give improved accuracy compared to the first order elements as they avoid the phenomenon known as locking.

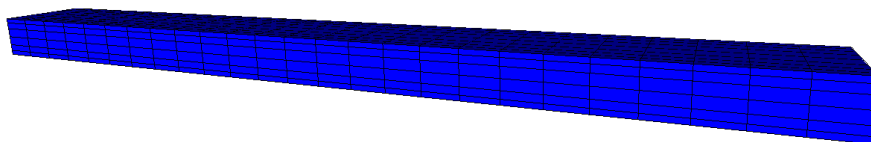


Figure 7.1: The mesh used in the computations

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried in steady-state in 3-dimensional Cartesian coordinates.

```
Model
  Setup
    Simulation Type = Steady state
    Steady state max. iter = 1
```

In the Equation section we choose the relevant equations which in this case only includes the Linear elasticity equation which solves the problem according to linear elastic theory. We also want to compute the stresses as a post-processing step. For the linear system solvers we change the default settings in order to obtain a better convergence in this case. As the equation is fully linear we also eliminate the non-linear iteration loop.

```
Model
  Equation
    Name = Elasticity
    Apply to Bodies = Body 1
    Linear elasticity
      Active = on
      Calculate Stresses = on
    Edit Solver Setting
      Linear System
        Method = Iterative / GCR
        Preconditioning = ILU1
      Nonlinear system
        Max. iterations = 1
    Apply
  Add
  OK
```

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such as Young's modulus and Poisson ratio.

```
Model
  Material
    Name = Pine
    General
      Density = 550
    Linear Elasticity
      Youngs Modulus = 10.0e9
      Poisson ratio = 0.37
    Apply to Bodies = Body 1
  Add
  OK
```

In this case there is a body force i.e. the gravity acting on the beam. We assume that the gravity points to the negative y direction.

```
Model
  BodyForce
    Name = Gravity
    Linear Elasticity
      Force 2 = $ -9.81 * 550
    Apply to Bodies = Body 1
```

```
Add
OK
```

Here we use a MATC expression for computing the volume force. This expression is constant and is computed when the command file is interpreted.

Convergence should be obtained with the default initial condition i.e. zero for all fields, hence no initial condition is applied.

The first boundary condition fixes the beam rigidly at the wall. The second boundary condition distributes the load of 2000 N uniformly on the area of $5.0e-3 \text{ m}^2$.

```
Model
BoundaryCondition
  Name = Wall
  Linear elasticity
    Displacement 1 = 0.0
    Displacement 2 = 0.0
    Displacement 3 = 0.0
Add
New

Name = Mass
Linear elasticity
  Force 2 = -4.0e5
Add
```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

```
Model
Set boundary properties
  Choose the wall end of the beam -> set boundary condition Wall
  Choose the other end of the beam -> set boundary condition Mass
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
Generate
Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case.

```
File
Save Project
```

After we have successfully saved the files we may start the solver

```
Run
Start solver
```

The simulation may take a minute or so depending on the speed of the processor. This time the convergence monitor does not have a meaningful output since of the different steps only one is related to the actual solution and the six other ones to the computation of stresses with the Galerkin method.

Results

When there are some results to view we may start the postprocessor, this time we use Paraview.

```
Run
  Start Paraview
```

Choose the displacement as the color field to plot. The maximum displacement is 6.36 cm You may also choose various stress components, or the von Mises stress. Also you may choose to see Surface With Edges to visualize the mesh also. The resulting picture is shown in Fig 7.2

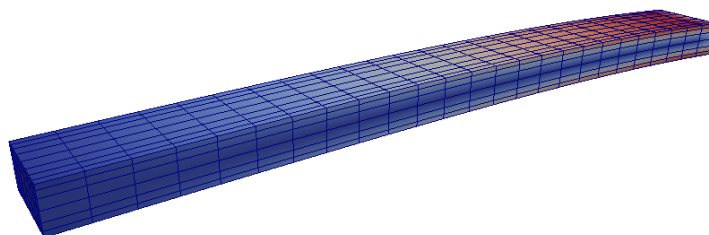


Figure 7.2: The displaced shape of the elastic beam colored with the von Mises stresses.

Note that the displacements are so large that the assumption of linearity may be severely questioned. When further increasing the loading one should resort to a solver that is able to catch the geometric non-linearities – the ElasticSolver.

Extra task: Gravity in x direction

The beam should be more rigid if the beam is oriented differently. For that aim, change the direction of gravity to orient in the negative x . Change the body force

```
Model
  BodyForce
    Linear Elasticity
      Force 1 = $ -9.81*550
    Update
  OK
```

and the boundary condition

```
Model
  BoundaryCondition
    Linear elasticity
      Force 1 = -4.0e5
    Update
  OK
```

The rigidity should scale as dh^3 and hence the maximum displacement should be reduced roughly to one quarter of the original.

Tutorial 8

Non-linear elasticity equation – 3D – Loaded elastic curve

Directory: ElasticHookNonlinear

Solvers: ElasticSolve

Tools: ElmerGUI

Dimensions: 3D, Transient

Author: Peter Råback

Case definition

An elastic U-shaped cylinder is pressed from its ends such that the object faces large displacements. The material properties of stainless steel are used. Problem is to gradually increase the displacement and visualize the increase of stresses. The problem requires the use of a solver capable of dealing with large displacements.

Solution procedure

The definitions for the relevant equation are not loaded into ElmerGUI by default. Hence, one needs to load these before starting the simulations.

File

Definitions

Append -> nonlinearelasticity.xml

The additional definitions should reside in the directory `edf-extra` within the distribution.

The mesh is given in ElmerGrid format in file `u_turn.grd`, load this file.

File

Open -> u_turn.grd

You should obtain your mesh and may check that it consists of 12288 trilinear elements and 13585 nodes.

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried as a transient problem in 3-dimensional Cartesian coordinates. The inertial forces are of no importance so we could as well scan through a set of steady state loading conditions. We choose the time step such that the total time being simulated is 1 s. We assume that the original mesh (with diameter 0.6 is given units of cm. Hence we need to scale the system down to work in SI units of meters.

Model

Setup

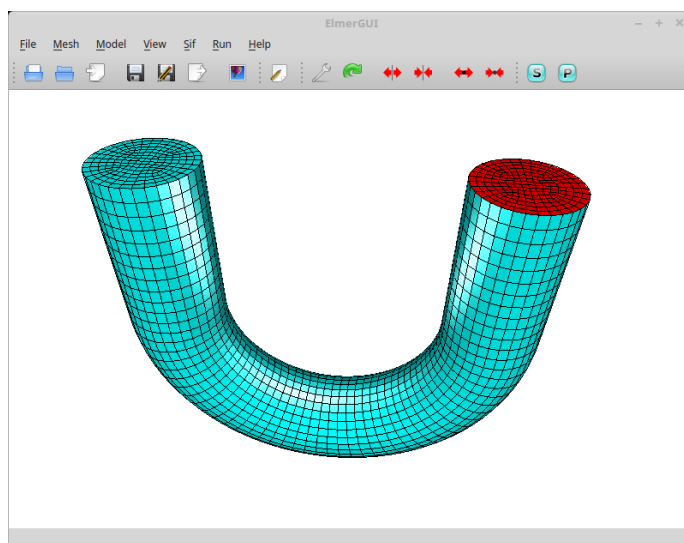


Figure 8.1: The mesh used in the computations as shown in ElmerGUI.

```
Simulation Type = Transient
Timestep Sizes = 0.05
Timestep Intervals = 20
Coordinate Scaling = 0.01
```

In the Equation section we choose the relevant equations which in this case only includes the `Nonlinear elasticity` equation. We have just one body and therefore its easy to assign the Equation and Material to it directly. We can use linear system solvers and we are happy to use the defaults. One may however, try out different preconditioners (`ILU1,...`) or more efficient iterative methods (`BiCGStabl`), for example.

```
Model
Equation
  Name = Elasticity
  Apply to Bodies = 1
  Nonlinear elasticity
    Active = on
    Edit Solver Settings
      Calculate Stresses = on
      Calculate Principal = on
  Add
  OK
```

Here we choose the stainless steel from the material library. You may click through the material parameters of the various solvers to ensure that the properties are indeed as they should be. Any consistent set of units may be used in Elmer. The natural choice is of course to perform the computations in SI units.

```
Model
Material
  Material library
    Austenitic stainless steel (AK Steel 201)
  Apply to Bodies = 1
  Add
  OK
```

There are no body forces and convergence should be easily obtained with the default initial condition i.e. zero for all fields. Hence we don't need to toggle these subitems.

We need to type now boundary conditions that we will later assign with the mouse to some surfaces. We set boundary conditions for both ends of the hook such that they close with respect to each other with time. The distance travelled in 1 s will be set to 0.006 m i.e. to same as the radius. The other displacement components are set to zero. To type in the multiline expressions for the boundary condition just press Enter in the Displacement 1 checkbox. Note that the semicolon is an alternative separator to line break.

```
Model
  BoundaryCondition
    Name = MovingRight
    Nonlinear elasticity
      Displacement 1 = Variable "time"
      Real MATC "0.006*tx"
      Displacement 2 = 0.0
      Displacement 3 = 0.0
    Add
  New

  Name = MovingLeft
  Nonlinear elasticity
    Displacement 1 = Variable "time"
    Real MATC "-0.006*tx"
    Displacement 2 = 0.0
    Displacement 3 = 0.0
  Add
```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

```
Model
  Set boundary properties
    Choose negative x end -> set boundary condition "MovingRight"
    Choose positive x end -> set boundary condition "MovingLeft"
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
  Generate
  Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case.

```
File
  Save Project
```

After we have successfully saved the files we may start the solver

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. As there are 20 time steps the convergence will be shown for each time step. You can start looking at the results already after a few time steps. The whole simulation takes a few minutes.

```
Run
  Start ParaView
```

In reality the material could break at some point. Here we have assumed a linear material law even though the geometric displacement makes the equation non-linear.

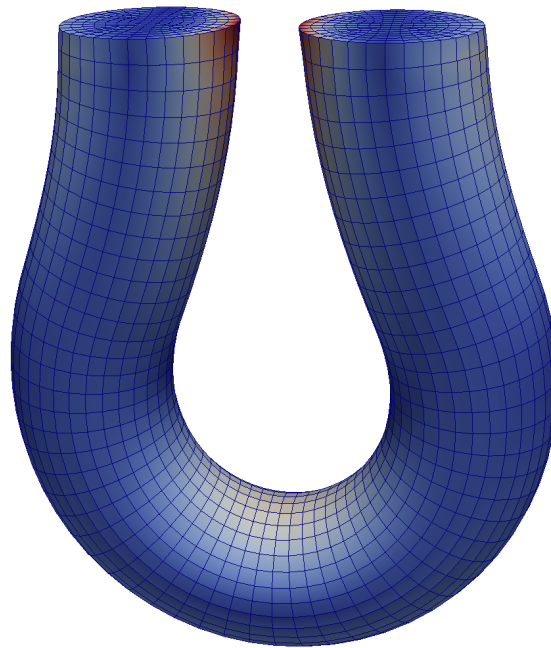


Figure 8.2: The final state of the deformed curve. Color shows the resulting von Mises stresses.

Alternative task: Rotating square profile

You may perform the tutorial with an alternative geometry: `square_profile.grd`. Follow almost the same logic except now we rotate the ends of the beam.

We need to enforce rigid body rotations to the ends of the mesh. The following types of Dirichlet conditions need to be implemented

$$\begin{aligned} u_x &= (\cos(\phi) - 1)x - \sin(\phi)y \\ u_y &= (\cos(\phi) - 1)y + \sin(\phi)x \end{aligned} \quad (8.1)$$

These can be set using the following MATC function

```
Displacement 1 = Variable "time, Coordinate"
Real MATC "(cos(tx(0)*pi)-1.0)*tx(1)-sin(tx(0)*pi)*tx(2)"
Displacement 2 = Variable "time, Coordinate"
Real MATC "(cos(tx(0)*pi)-1.0)*tx(2)+sin(tx(0)*pi)*tx(1)"
```

Note that the argument to MATC is always called `tx` and it holds the parameters in the given order. In this case component "0" refers to `time`, component "1" to `Coordinate 1` (i.e. `x`), and component "2" to `Coordinate 2` (i.e. `y`). The multiplier for the trigonometric functions is π which means that a revolution of 180 degrees is performed in 1 second.

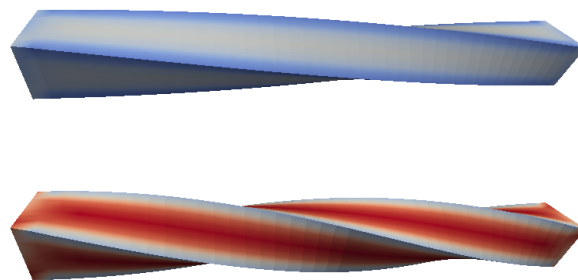


Figure 8.3: A square profile being rotated from one end by 90 and 180 degrees. Color shows the von Mises stresses.

Tutorial 9

Smitc solver – 2D – Deflection of a linear elastic plate

Directory: ElasticPlateLinear

Solvers: SmitcSolver

Tools: ElmerGUI

Dimensions: 2D, Eigenmode

Author: Peter Råback

Problem description

This tutorial demonstrates how to use the Smitc solver to solve small deflections of plates. The Smitc solver is for elastic linear plates and uses the theory of Reissner and Mindlin.

The case under investigation is a L-shaped steel plate under pressure. The plate is shown in figure 9.1. The longer sides have the length of 2 m and the shorter 1 m . So the area of the plate is 3 m^2 . The plate has a thickness of 1 cm . We assume that on the plate there is about 15300 kg of sand. The sand is uniformly distributed on the plate and the sand stays uniformly distributed even if the plate undergoes small deflection. The sand exerts to the plate a pressure of 50000 Pa . The plate is clamped from all sides meaning that both deflection and rotation are zero on all edges.

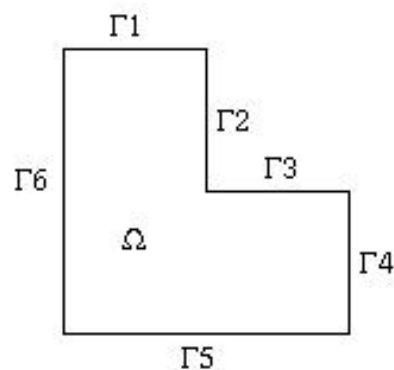


Figure 9.1: The geometry of plate and the numbering of edges.

For more details on the solver we refer to the documentation of Smitc solver in the Elmer Models Manual.

Solution procedure

Start ElmerGUI from command line or by clicking the icon in your desktop. Here we describe the essential steps in the ElmerGUI by writing out the clicking procedure. Indentation generally means that the selections are done within the window chosen at the higher level.

Before we can start the set-up we should make sure that the menus for Smitc solver are present. If not, they may be found in file

```
$ELMERHOME/bin/edf-extra/elasticplate.xml
```

To load these definitions do the following

```
File
  Definitions
    Append -> choose the file
```

To see what kind of new menu structures were loaded you may play around with viewer collapsing and opening. Note that if you want to load an existing project you should load the xml-definitions that were used in creating the project. Therefore it may be best to place all actively used menu definitions in directory

```
$ELMERHOME/bin/edf
```

When the menu structures for plate solver are there then we are ready to continue. The first thing to do is create a mesh with ElmerGrid. The definition of mesh is in the file `simple_plate.grd`. The mesh is about uniform and consist of 1000 linear square elements.

```
File
  Open -> simple_plate.grd
```

You should obtain a L shaped figure with 6 sides. You may check in the `Model summary...` window that it consists of 1082 nodes and 1008 surface elements. If the mesh was successfully imported your window should look something like figure 9.2.

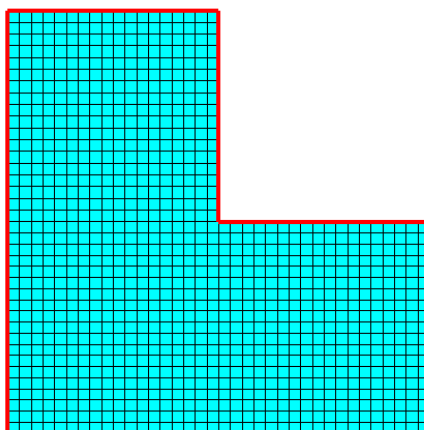


Figure 9.2: The plate mesh

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc.

The simulation uses 2D Cartesian geometry. The simulation is not time dependent i.e. Steady State. There are no coupled solvers so only one iteration is needed.

```

Model
  Setup
    Simulation Type = Steady state
    Steady state max. iter = 1
  Apply

```

In the equation section we choose the relevant equations and parameters related to their solution. When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and therefore its easier to assign the Equation and Material to it directly.

```

Model
  Equation
    Add
      Name = Elastic Plate
      Apply to bodies = 1
      Elastic Plates
        Active = on
    Add
  OK

```

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such heat Young's modulus. As our problem is academic in nature we choose some simple ideal parameters, similar to steel, but data from material database could also be used instead.

```

Model
  Material
    Add
      Name = Ideal
      Apply to bodies = 1
      General
        Density = 7800.0
      Elastic Plates
        Youngs Modulus = 209.0e9
        Poisson ratio = 0.3
        Thickness = 1.0e-2
        Tension = 0.0
    Add
  OK

```

A Body Force represents the right-hand-side of a equation i.e. external forces. In Body Force block we give the equations right hand side. It is the uniform pressure from the weight of the sand and it is the same constant in every point.

```

Model
  BodyForce
    Name = Pressure
    Apply to Bodies = Body 1
    Elastic Plates
      Pressure = 5.0e4
    Add
  OK

```

In this case all the boundaries are rigidly fixed we set all the components of the solution field to be zero. The 1st component is the displacement in the normal direction while the 2nd and 3rd components are actually the components of rotation vector

```
Model
  BoundaryCondition
    Add
      Elastic Plates
        Deflection 1 = 0.0
        Deflection 2 = 0.0
        Deflection 3 = 0.0
      Name = Fixed
      Apply to boundaries = 1 2 3 4 5 6
    Add
      OK
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
  Generate
  Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. In saving the project all the necessary files for restarting the case will be saved to the destination directory.

```
File
  Save Project
```

After we have successfully saved the files we may start the solver

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. In this case there is just one iteration and thus no curve appears. The problem is solved in few seconds.

Results

To view the results start Paraview

```
Run
  Paraview
```

and select the 1st component of the deflection field (confusingly named the x-component).

Currently you would use Paraview, ElmerVTK, or something that can read the VTU files. Here the results are shown from Paraview.

Result for displacement (deflection x) is shown in figure 9.3.

Note that the 1st component of Deflection is the displacement to normal direction whereas the 2nd and 3rd components are the x- and y-components of rotation vector, and are shown in figure 9.4.

Extra task

You may test the effect of pre-stressing by altering the Tension material parameter.

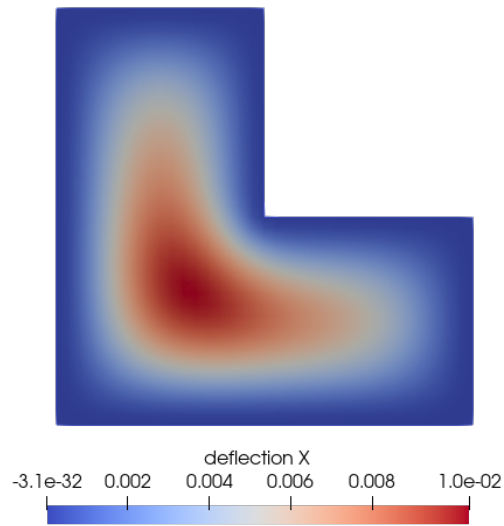


Figure 9.3: The deflection of the plate.

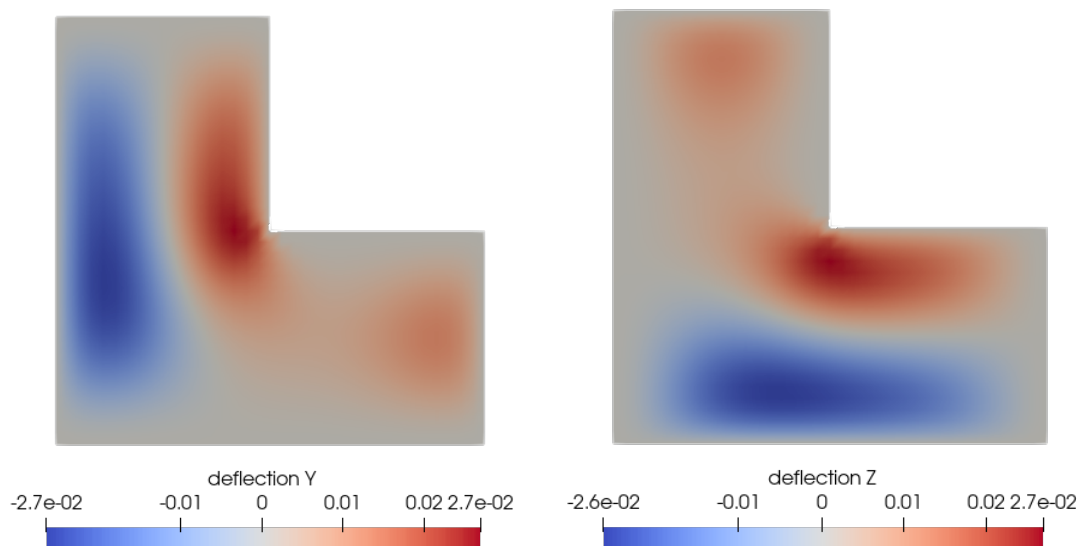


Figure 9.4: The rotation of the plate in x and y components.

Tutorial 10

Smitc solver – 2D – Eigenmodes of an elastic plate

Directory: ElasticPlateEigenmodesGUI

Solvers: SmitcSolver

Tools: ElmerGUI

Dimensions: 2D, Eigenmode

Author: Peter Råback

Problem description

For thin elastic structures it is often advisable to use dimensionally reduced models i.e. study plates or shells. In this tutorial we compute the few lowest eigenmodes of an elastic plate. Our geometry is a simple pentagon which (compared to a square) eliminates some of the trivial symmetries. The pentagon is rigidly fixed at all boundaries.

For more details on the solver we refer to the documentation of Smitc solver in the Elmer Models Manual.

Solution procedure

Start `ElmerGUI` from command line or by clicking the icon in your desktop. Here we describe the essential steps in the ElmerGUI by writing out the clicking procedure. Tabulation generally means that the selections are done within the window chosen at the higher level.

Before we can start the set-up we should make sure that the menus for Smitc solver are present. If not, they may be found in file

```
$ELMERHOME/bin/edf-extra/elasticplate.xml
```

To load these definitions do the following

```
File
  Definitions
  Append -> choose the file
```

To see what kind of new menu structures were loaded you may play around with viewer collapsing and opening. Note that if you want to load an existing project you should load the xml-definitions that were used in creating the project. Therefore it may be best to place all actively used menu definitions in directory

```
$ELMERHOME/bin/edf
```

When the menu structures for plate solver are there then we are ready to continue. The mesh is given in 2d netgen format in file `pentagon.grd` in the samples directory of ElmerGUI, load this file.

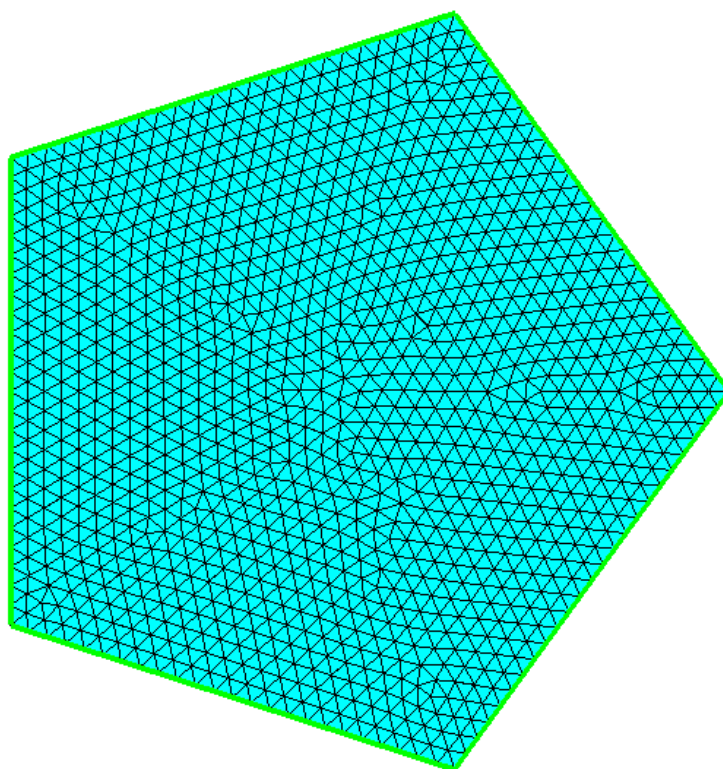


Figure 10.1: The finite element mesh in ElmerGUI

File

```
Open -> pentagon.in2d
```

You should obtain a pentagon consisting of 5 triangles. To increase the number of elements change the parameters passed on to the nglib library by going to

Mesh

```
Configure
```

```
nglib / Max H: 0.05
```

You may check in the Model summary window that it consists of 1199 nodes and 2276 linear triangles. If the mesh was successfully imported your window should look something in figure 10.1.

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried out in 2-dimensional Cartesian coordinates and in steady-state (also used for eigenmodes). Only one steady-state iteration is needed as the case is linear.

Model

```
Setup
```

```
Simulation Type = Steady state
```

```
Steady state max. iter = 1
```

```
Apply
```

In the equation section we choose the relevant equations and parameters related to their solution. When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and therefore its easier to assign the Equation and Material to it directly.

For the solver setting we need to activate the eigen mode computation. We also choose the direct umfpack solver which for small 2D problems often performs great.

```
Model
  Equation
    Add
      Name = Plate Equation
      Apply to bodies = 1
      Elastic Plates
        Active = on
        Edit Solver Settings
          Solver Specific Options
            Eigen Analysis = on
            Eigen System Values = 10
          Linear System
            Direct = on
            Umfpack
    Add
  OK
```

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such heat Young's modulus. As our problem is academic in nature we choose some simple ideal parameters but data from material database could also be used instead.

```
Model
  Material
    Add
      Name = Ideal
      Apply to bodies = 1
      General
        Density = 1000.0
      Elastic Plates
        Youngs Modulus = 1e9
        Poisson ratio = 0.3
        Thickness = 0.001
        Tension = 0.0
    Add
  OK
```

A Body Force represents the right-hand-side of a equation i.e. external forces. In eigenmode analysis no body forces are used. Nor are any Initial conditions required.

In this case all the boundaries are rigidly fixed we set all the components of the solution field to be zero. The 1st component is the displacement in the normal direction while the 2nd and 3rd components are its derivatives in x and y directions.

```
Model
  BoundaryCondition
    Add
      Elastic Plates
        Deflection 1 = 0.0
        Deflection 2 = 0.0
        Deflection 3 = 0.0
      Name = Fixed
      Apply to boundaries = 1 2 3 4 5
    Add
  OK
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
  Generate
  Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. In saving the project all the necessary files for restarting the case will be saved to the destination directory.

```
File
  Save Project
```

After we have successfully saved the files we may start the solver

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. In this case there is just one iteration and thus no curve appears.

Results

The resulting eigenvalues are shown in table 10.1. Note that some eigenmodes are degenerated but as the finite element mesh is not perfectly symmetric there will be minor differences in the eigenvalues.

Table 10.1: Ten lowest eigenvalues for the pentagon plate

No	ω^2
1	18.9
2,3	81.3
4,5	214.5
6	281.1
7, 8	472.5
9, 10	621.0

Note: if you face problems in the solution phase and need to edit the setting, always remember to save the project before execution.

To view the results we here start Paraview

```
Run
  Paraview
```

and select the 1st component of the deflection field (confusingly named the x-component). If one chose for vtU output the mode `Eigen Analysis = True` then each file includes one eigenmode. Then the eigenmodes may be treated as time steps. In figure 10.2 some of the lowest eigenmodes are depicted.

Extra task

You may test the effect of pre-stressing by altering the Tension material parameter.

There are other similar geometries that you could use i.e. `hexagon.in2d`, `heptagon.in2d`, `octagon.in2d`. When the number of vertices is increased the eigenvalues should slightly decrease.

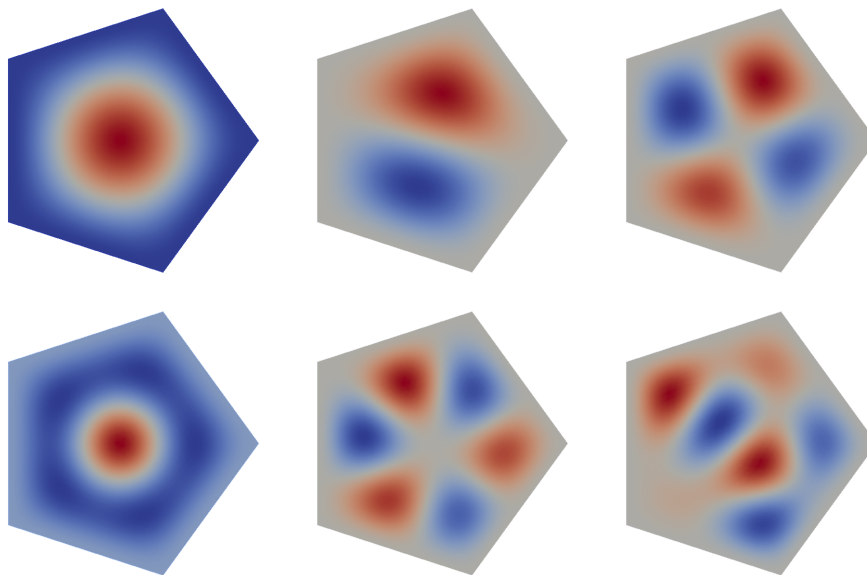


Figure 10.2: The 1st, 2nd, 4th, 6th, 7th and 9th eigenmode of the plate. The displacement in the normal direction (first component) is shown.

Tutorial 11

Electrostatic equation – Computation of fringe capacitance

Directory: FringeCapacitance

Solvers: StatElecSolver

Tools: ElmerGUI

Dimensions: 2D, Steady-state

Author: Peter Råback

Case definition

This case presents solving the Laplace equation for electric potential.

$$-\nabla \cdot \varepsilon \nabla \phi = 0 \quad \in \Omega \quad (11.1)$$

where the electric potential ϕ is given at conducting surfaces. This is a standard type of equation with many variants in physics, electrostatics being just one of them. From the solution one may calculate derived fields, and capacitance which is obtained from the total electric energy

$$E = \frac{1}{2} \int \varepsilon |\nabla \phi|^2 d\Omega \quad (11.2)$$

and the relation $E = CU^2/2$.



Figure 11.1: The geometry of the capacitor

The geometry studied is a 2D plate capacitor having the well known approximation for Capacitance $C_{app} = \varepsilon_r \varepsilon_0 A/d$. With the help of the simulation one may evaluate the fringe capacitance resulting from the end effects of the capacitor geometry. The measurements of the capacitor are 10×1 , and the distance

to ground is also 1. Defining the permittivity of vacuum to be $\epsilon_0 = 1$ the comparison to the analytical approximation becomes trivial since then then $C_{app} = 10$.

The derived fields in the StatElecSolver are computed by averaging the fields over elements – not using the Galerkin method which would provide optimal accuracy. The fields may, however, be sufficient for visualization purposes.

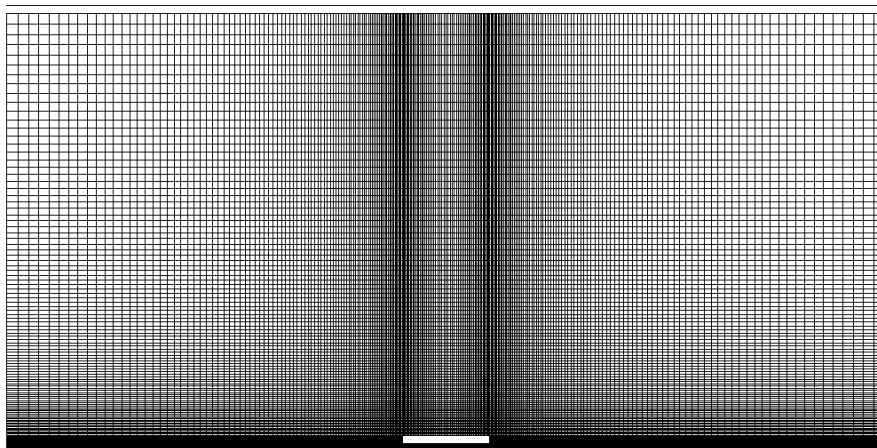


Figure 11.2: Computational mesh used in the simulation

Solution procedure

The definitions for the electrostatic equation are not loaded into ElmerGUI by default. Hence, one needs to load these before starting the simulations.

```
File
  Definitions
    Append -> electrostatics.xml
```

The additional definitions should reside in the directory `edf-extra` within the distribution. Moving the desired `xml` files to the `edf`-directory enables automatic loading of the definitions at start-up. By inspecting the definitions in the Elmer Definitions File editor one may inspect that the new definitions were really appended.

The mesh is given in ElmerGrid format in file `disc.grd`, load this file.

```
File
  Open -> disc.grd
```

You should obtain your mesh and may check that it consists of 30484 nodes and of 30348 bilinear elements.

```
Model
  Summary...
```

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. The steady-state simulation is carried out in 2-dimensional Cartesian coordinates. For convenience we also set ϵ equal to one.

```
Model
  Setup
    Simulation Type = Steady state
    Vacuum Permittivity = 1.0
```

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have only the electrostatics solver.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and therefore its easier to assign the Equation and Material to it directly.

In the solver specific options we want to activate some flags that are needed to invoke the computation of derived fields. For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1,...) or direct Umfpack solver, for example.

```
Model
  Equation
    Name = Electrostatics
    Apply to Bodies = 1
    Electrostatics
      Active = on
      Edit Solver Settings
        Solver specific options
          Calculate Electric Field = True
          Calculate Electric Energy = True
    Add
  OK
```

The Material section includes all the material parameters. In this case we only have the relative permittivity which we set to one.

```
Model
  Material
    Name = Ideal
    Electrostatics
      Relative Permittivity = 1.0
    Apply to Bodies = 1
    Add
  OK
```

We have two boundary conditions for the potential at the ground and at the capacitor. For other boundaries the do-nothing boundary results to zero flux over the boundary.

```
Model
  BoundaryCondition
    Name = Ground
    Electrostatics
      Potential = 0.0
    Add
  New

  Name = Capacitor
  Electrostatics
    Potential = 1.0
  Add
```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

```
Model
  Set boundary properties
    Choose Ground -> set boundary condition Ground
    Choose Capacitor -> set boundary condition Capacitor
```


For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
  Generate
  Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case.

```
File
  Save Project
```

After we have successfully saved the files we may start the solver

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. The equation is fully linear and hence only two iterations are needed – the second one just ensures that convergence of the non-linear level was really obtained. The norm of the solution should be 0.3055.

When the solution has finished we may start the postprocessor to view some results.

```
Run
  Start ParaView
```

Results

From the output of the simulation one may see that the capacitance in this case was 13.697 compared to the analytical estimate of 10. Hence the fringe capacitance in this case increases the capacitance by 37 %.

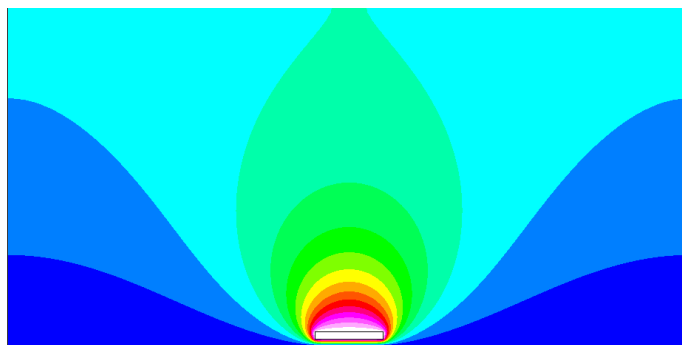


Figure 11.3: The electrostatic potential

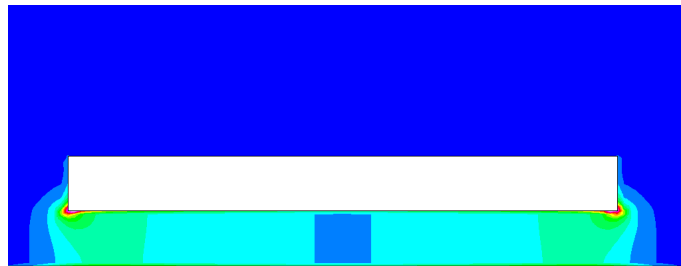


Figure 11.4: The electrostatic energy density, a close-up

Tutorial 12

Electrostatic equation – Capacitance of two balls

Directory: CapacitanceOfTwoBalls

Solvers: StatElecSolver

Tools: netgen,ElmerGUI

Dimensions: 3D, Steady-state

Author: Peter Råback

Case definition

This case presents the solution of the capacitance of perfectly conducting balls in free space. A voltage difference between the balls results to electric charge being introduced to the system. The balls have also self-capacitance that comes from the voltage difference with the far field. Therefore a symmetric capacitance matrix with of size 2×2 needs to be solved. The capacitances may be computed from two different voltage configurations. For both the electrostatic equation is solved automatically.

The problem does not have an analytical solution in a closed form. However, the cross-capacitance between the balls may be approximated from the series solution in ¹:

$$C_{12} = 4\pi\epsilon \frac{a^2}{d} \left(1 + \frac{a^2}{d^2 - 2a^2} + \frac{a^4}{d^4 - 4d^2a^2 + 3a^4} + \dots \right) \quad (12.1)$$

and the self-capacitance from

$$C_{10} = C_{20} = 4\pi\epsilon a \left(1 - \frac{a}{d} + \frac{a^2}{d^2 - a^2} - \frac{a^3}{d^3 - 2da^2} + \dots \right) \quad (12.2)$$

Let's mark $\tilde{C} = C/\epsilon$. In this case $\tilde{C}_{12} \approx 1.191$ and $\tilde{C}_{10} \approx 5.019$. Unfortunately the error bounds are not given.

In this particular case the balls are assumed to have a radius of $a = 0.5$ and they are placed at distance $d = 2$ apart from each other (measured from the ball origins).

Meshing with Netgen (Optional)

If you have Netgen installed and running, then you can generate the mesh before starting ElmerGUI, following the below instructions.

If you don't have Netgen available, don't worry, an Elmer mesh has been provided in the 'Capacitance-OfTwoBalls' subfolder under 'tutorials-GUI-files'. Just load the ElmerGUI project in that subfolder, and it will load the mesh for you. After loading the mesh, skip ahead to the next section, 'Solution Procedure'.

¹TUW-101191, PHD Dissertation by Christoph Wasshuber, About Single-Electron Devices and Circuits, Institut für Mikroelektronik, 1997, Appendix A.3, Capacitance of Two Spheres, <https://www.iue.tuwien.ac.at/phd/wasshuber/>

If you have Netgen installed, then meshing is performed with the graphical user interface of Netgen. Netgen creates tetrahedral quality meshes and provides a native output for Elmer. At the time of writing this tutorial the quadratic elements had some problems with numbering but these should not affect the linear elements.

The file is given as netgen geometry format in file `TwoBallsInBall.geo`. The geometry definition includes the two smaller balls inside a bigger ball. Ultimately the bigger ball would be infinitely large. As this is impossible here we choose a modest radius of 5. The larger this value, the better the far-field approximation of the electrostatic solution is.

The content of the file is given below:

```
#
# a large ball with two smaller balls cut off
#
algebraic3d
solid smallballs = sphere (-1.0, 0.0, 0.0; 0.5)
                  or sphere (1.0, 0.0, 0.0; 0.5);
solid bigball = sphere (0.0, 0.0, 0.0; 5.0);
solid rest = bigball and not smallballs;
tlo rest -col=[0,0,1] -transparent;
```

Open the file and apply the default meshing. In this example two consecutive uniform refinements were performed (choose `Refine Uniform` under `Refinement`) so that the final mesh consisted of 41 693 nodes and 238 976 linear tetrahedrons.

To save the mesh first choose under `File` the `Export Filetype` to be `Elmer`. Then choose `Export Mesh` and save the mesh into a suitable directory to be opened by `ElmerGUI`.

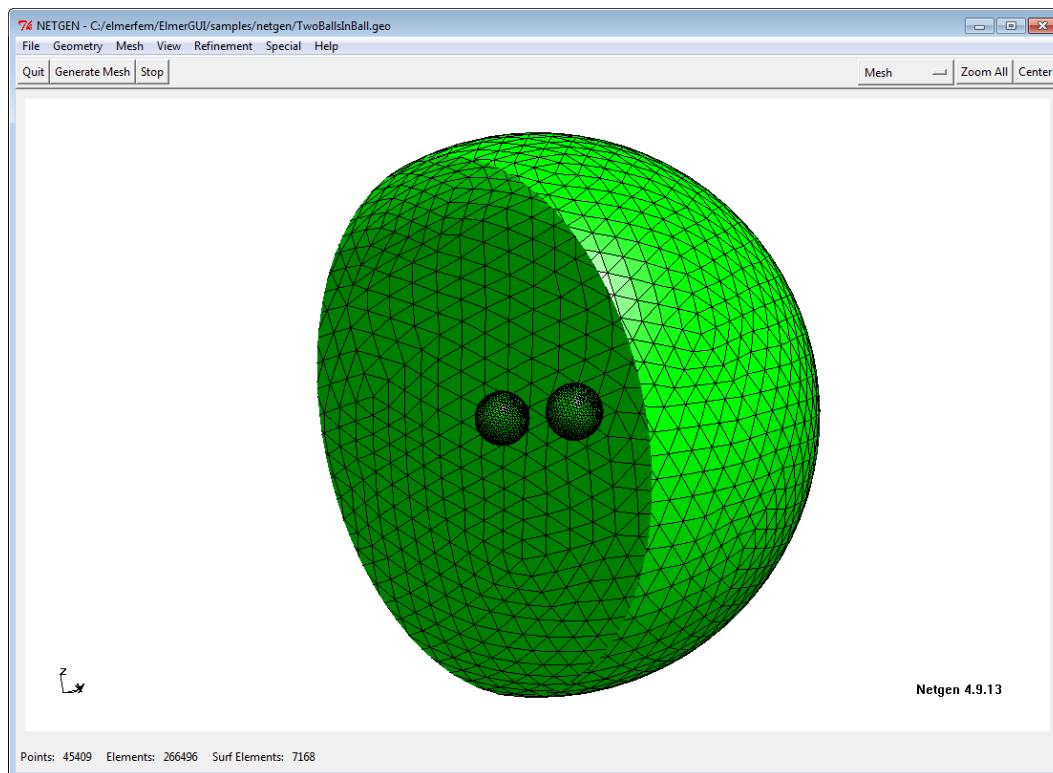


Figure 12.1: Surface mesh for the two inner balls as seen in Netgen

The order of the mesh using nodal elements may be increased by `ElmerGrid`. Assuming the mesh would reside in directory `meshlin` a mesh consisting of quadratic elements may be performed with the following command:

```
ElmerGrid 2 2 meshlin -increase -out meshquad
```

This will maintain the number of elements but the number of nodes will, in this case, increase to 359 009.

Solution procedure

The definitions for the electrostatic equation may not have been loaded into ElmerGUI by default. If this is the case one needs to load these before starting the simulations.

```
File
  Definitions
    Append -> electrostatics.xml
```

The additional definitions should reside in the directory `edf-extra` within the distribution. Moving the desired `xml` files to the `edf`-directory enables automatic loading of the definitions at start-up. By inspecting the definitions in the Elmer Definitions File editor one may inspect that the new definitions were really appended.

The mesh is already created, load it from the directory that was created above.

```
File
  Load Mesh -> mesh
```

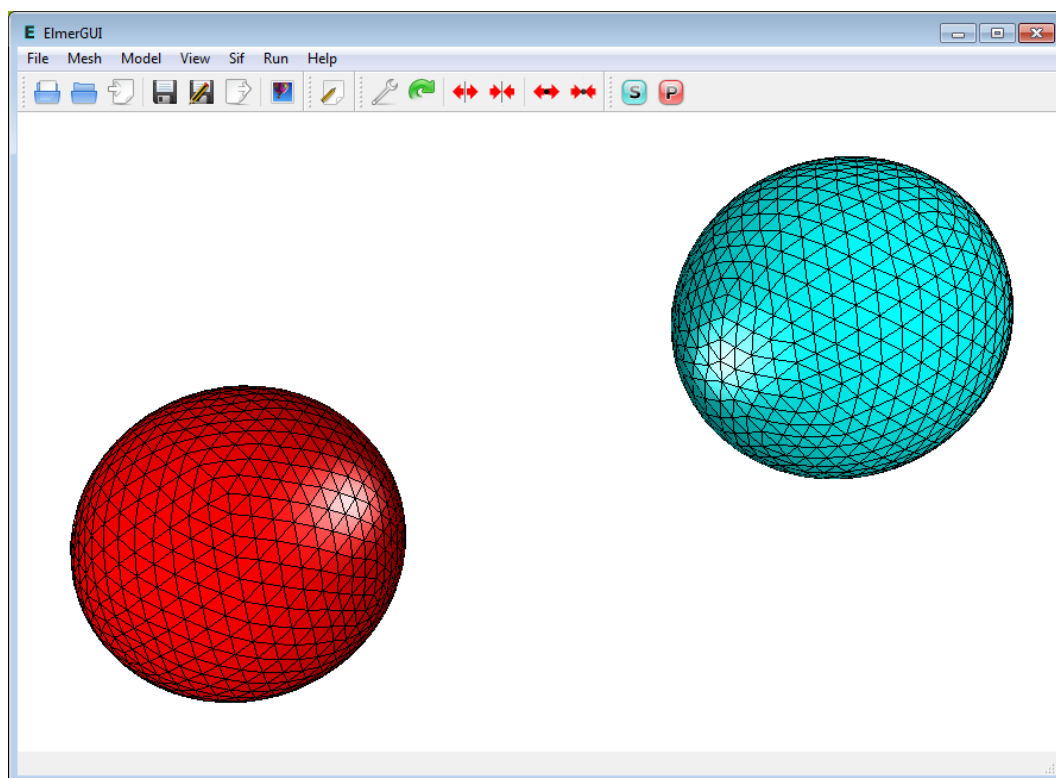


Figure 12.2: The mesh with one highlighted ball as seen in ElmerGUI

The display in ElmerGUI upon loading the mesh will show the outside of the large ball. To view the inner two smaller balls, double click on the outside of the large ball, then click on 'Hide/show selected'.

```
View
  Hide/show selected   Ctrl+H
```

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. The steady-state simulation is carried out in 3-dimensional Cartesian coordinates. For convenience we also set the permittivity of vacuum ε_0 equal to one. This makes it easier to compare the results to the analytical expressions.

```
Model
  Setup
    Simulation Type = Steady state
    Vacuum Permittivity = 1.0
```

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have only the electrostatics solver.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and therefore its easier to assign the Equation and Material to it directly.

In the solver specific options we want to activate some flags that are needed to invoke the computation of derived fields. For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1,...) or direct Umfpack solver, for example.

```
Model
  Equation
    Name = Electrostatics
    Apply to Bodies = 1
    Electrostatics
      Active = on
      Edit Solver Settings
        Solver specific options
          Calculate Capacitance Matrix = True
          Calculate Electric Field = True
          Calculate Electric Energy = True
    Add
    OK
```

The Material section includes all the material parameters. In this case we only have the relative permittivity ε_r which we set to one.

```
Model
  Material
    Name = Ideal
    Electrostatics
      Relative Permittivity = 1.0
    Apply to Bodies = 1 2
    Add
    OK
```

We have two boundary conditions for the potential at the ground and at the capacitor. For other boundaries the do-nothing boundary results to zero flux over the boundary.

```
Model
  BoundaryCondition
    Name = Farfield
    Electrostatics
      Electric Infinity BC = True
    Add
    New
```

```
Name = CapBody1
Electrostatics
  Capacitance Body = 1
Add
New
```

```
Name = CapBody2
Electrostatics
  Capacitance Body = 2
Add
```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

```
Model
  Set boundary properties
    Choose Outer sphere -> set boundary condition Farfield
    Choose one inner sphere -> set boundary condition CapBody1
    Choose the other inner sphere -> set boundary condition CapBody2
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
  Generate
  Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case.

```
File
  Save Project
```

After we have successfully saved the files we may start the solver

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. The equation is fully linear and hence only two iterations are needed – the second one just ensures that convergence of the non-linear level was really obtained. The norm of the solution should be 0.36356324.

When the solution has finished we may start the postprocessor to view some results.

```
Run
  Start ParaView
```

Results

The essential result of this case are the values of the capacitance matrix. In this case $\tilde{C}_{12} \approx 1.691$ and $\tilde{C}_{10} \approx 5.019$. For linear elements the obtained figures are 1.6983, 5.0793 and 5.0812, for quadratic Lagrange elements 1.6641, 5.0340 and 5.0340, respectively, and finally for quadratic p-elements 1.6856, 4.9863 and 4.9884.

The values are rather satisfactory with a difference less than 2% from the series approximation.

Note that the derived fields in the StatElecSolver are computed by averaging the fields over elements – not using the Galerkin method which would provide optimal accuracy. To get optimal accuracy, use FluxSolver, for example.

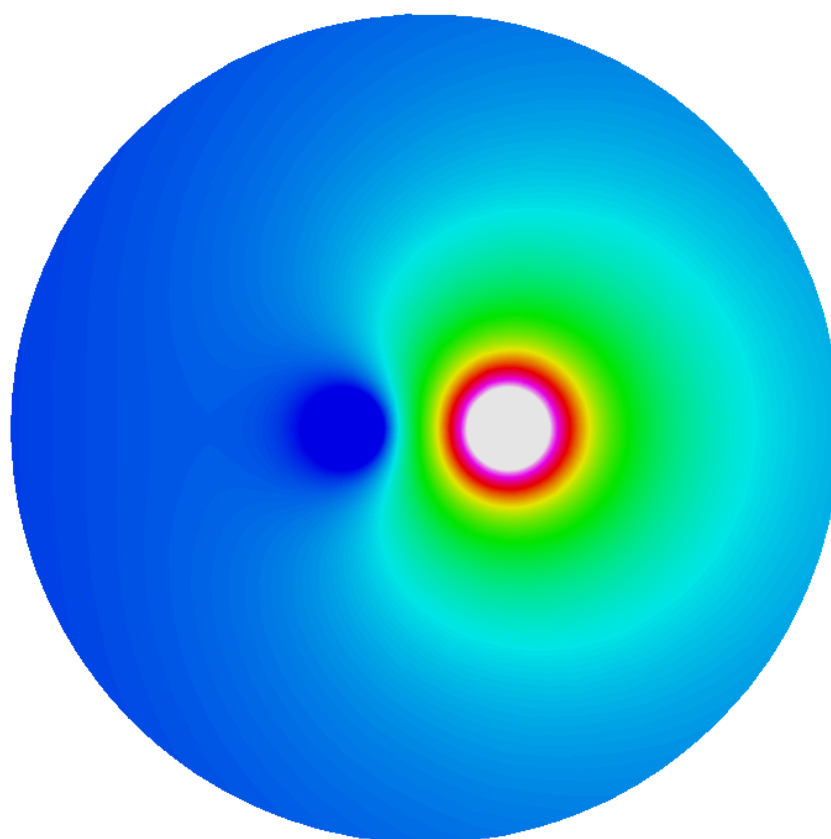


Figure 12.3: The electrostatic potential on the clipping plane. This is the latter of the two symmetric configurations where the unit voltage is applied to one ball and zero voltage to the other, respectively.

Tutorial 13

Electrostatic equation – Capacitance of perforated plate

Directory: CapacitanceOfPerforatedPlate

Solvers: StatElecSolver

Tools: ElmerGUI

Dimensions: 3D, Steady-state

Author: Peter Råback

Case definition

This case presents solving the Poisson equation for electric potential and calculating appropriate derived quantities, such as capacitance, based on the result. The geometry under study is a perforated plate.

The shape of the holes is assumed to be square with size $3 \times 3 \text{ mm}^2$. The holes cover the other plate uniformly so that the size of each unit cell is $10 \times 10 \text{ mm}^2$. The thickness of the plate is assumed to be 1.5 mm and the distance from the reference plate 1.0 mm. The plate may be assumed to be infinitely large. Due to symmetry considerations it suffices to study only one quarter of a unit cell.

The results may be compared to the ideal plate capacitor without holes. For a plane capacitor, the capacitance is

$$C = \varepsilon_r \varepsilon_0 \frac{A}{d}, \quad (13.1)$$

where ε_r is the relative permittivity, ε_0 is the permittivity of vacuum, A is the area of a capacitor plate, d is the separation of the capacitor plates, and Φ is the potential difference between the plates. For the case under study we get an approximation $C = 221.36 \text{ fF}$.

Preliminaries

The definitions for the electrostatic equation might not have been loaded into ElmerGUI by default. Check the `Model/Equation` menu to verify the situation. If electrostatics is not present one needs to load definitions for it before starting the simulations.

File

Definitions

Append -> electrostatics.xml

The additional definitions should reside in the directory `edf-extra` within the distribution. Moving the desired `xml` files to the `edf`-directory enables automatic loading of the definitions at start-up. By inspecting the definitions in the `Elmer Definitions File` editor one may inspect that the new definitions were really appended.

Meshing

In this case meshing is performed using ElmerGrid format in file `hexhole.grd` and the ElmerGrid plugin within ElmerGUI. The default mesh is OK and therefore no modifications are needed by the user.

Elmer does not operate an any particular units but usually SI-units are preferred. We therefore choose to scale the problem with 0.001 so that these measurements will be in mm.

Load the mesh file.

File

```
Open -> hexhole.grd
```

You should obtain your mesh and may check that it consists of roughly of 33 159 nodes and of 29 610 linear hexahedral elements.

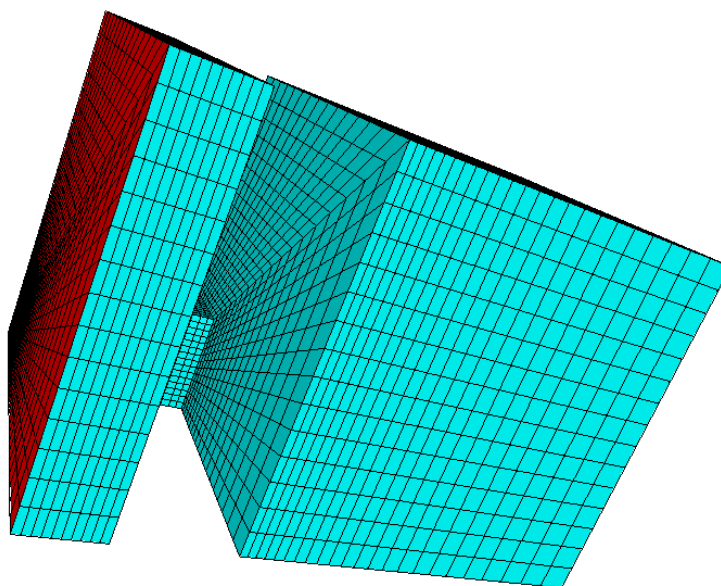


Figure 13.1: The mesh with one highlighted backplate as seen in ElmerGUI

Case setup

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. The steady-state simulation is carried out in 3-dimensional Cartesian coordinates. We want to work in mm so we need to scale the mesh with a factor 0.001.

Model

Setup

```
Simulation Type = Steady state
```

```
Coordinate Scaling = 0.001
```

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have only the electrostatics solver.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and therefore its easier to assign the Equation and Material to it directly.

In the solver specific options we want to activate some flags that are needed to invoke the computation of derived fields. For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1,...) or direct Umfpack solver, for example.

```
Model
Equation
  Name: Electrostatics
  Apply to Bodies: 1
  Electrostatics
    Active: on
    Edit Solver Settings
      Solver specific options
        Calculate Electric Field: True
        Calculate Electric Energy: True
  Add
  OK
```

The Material section includes all the material parameters. In this case we only have the relative permittivity ϵ_r which we could set to one. Alternatively, we can obtain it automatically by choosing the properties of air.

```
Model
Material
  Add
  Material library
    Air (room temperature)
  Apply to bodies: Body 1
  Add
  OK
```

We have two boundary conditions for the potential at the ground and at the capacitor. For other boundaries the do-nothing boundary results to zero flux over the boundary.

```
Model
BoundaryCondition
  Name: Ground
  Electrostatics
    Potential: 0.0
  Add
  New

  Name: Capacitor
  Electrostatics
    Potential: 1.0
  Add
```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

```
Model
Set boundary properties
  Choose the backplate -> set boundary condition Ground
  Choose the four pieces of the perforated plate -> set boundary condition Capacitor
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
  Generate
  Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case.

```
File
  Save Project
```

After we have successfully saved the files we may start the solver

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. The equation is fully linear and hence only two iterations are needed – the second one just ensures that convergence of the non-linear level was really obtained. The norm of the solution should be roughly 0.8846.

Numerical results

The numerical results obtained for capacitance and electric force are compared to those of a complete plane capacitor.

The results of the simulation as well as the comparison to the complete plane capacitor values are shown in Table 13.1.

Table 13.1: Comparison of numerical results to analytic values

relh	C(fF)	ratio
1.0	216.37	0.977
0.7	216.34	
0.5	216.32	

Visualization

When the solution has finished we may start the postprocessor to view some results. Here we assume that we chose the .vtu format for Paraview as output. The opening of Paraview can be done automatically from the ElmerGUI menu, or alternatively manually from Paraview.

```
File
  Open: case0001.vtu
  Apply
```

You may now choose the fields to be depicted etc. For more information on the use of of Paraview look at the documentation of the software. In figure 13.2 some examples of visualizations with Paraview are given.

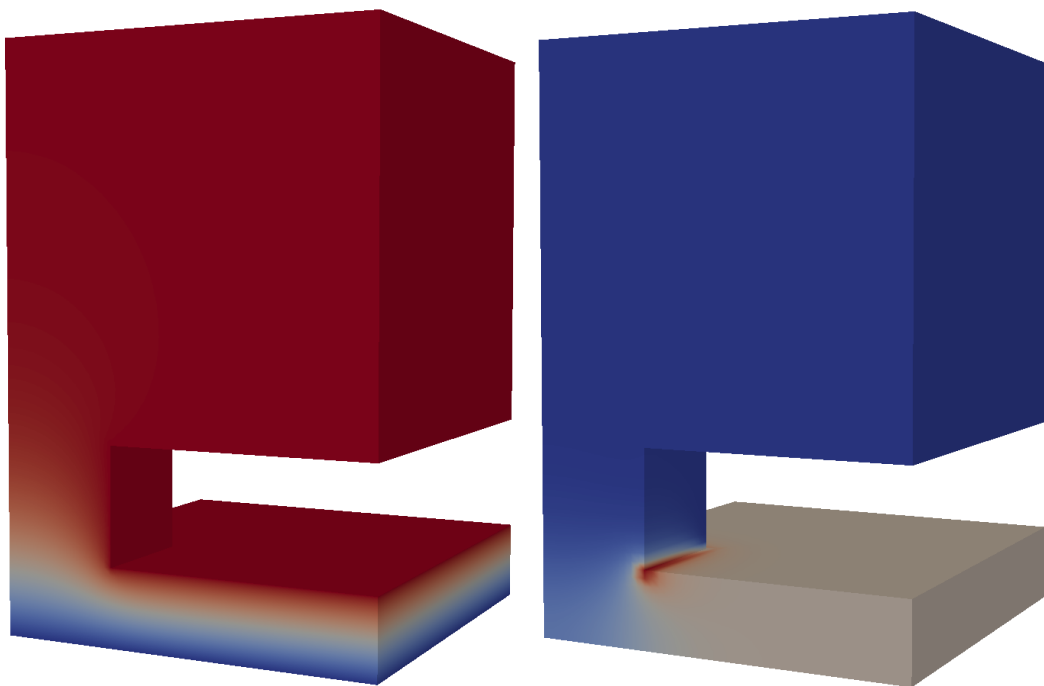


Figure 13.2: The electrostatic potential and the electric energy density of the quarter of a unit cell of a perforated plate capacitor visualized by Paraview.

Tutorial 14

Magnetic field induced by harmonic current in a wire

Directory: MagneticFieldWire

Solvers: WhitneyAVHarmonicSolver

Tools: ElmerGrid, ElmerGUI

Dimensions: 3D, Harmonic

Author: Peter Råback

Case definition

This case demonstrates the simplest case on how to utilize the edge element based solvers for computation of magnetic and electric fields.

Consider a simple copper wire with radius $R = 0.001$ m and length $R = 0.01$ m. A potential difference of 1 V/m is applied over the wire. We want to know the magnetic field strength around the wire.

For steady state we have a simple analytical solution for reference.

$$\vec{A} = \begin{cases} -\frac{I\mu_0}{2\pi} \ln(r/R)\vec{e}_z, & r > R \\ -\frac{I\mu_0}{4\pi R^2} (r^2 - R^2)\vec{e}_z, & r \leq R \end{cases} \quad (14.1)$$

resulting to a magnetic flux density

$$\vec{B} = \begin{cases} \frac{I\mu_0}{2\pi r}\vec{e}_\phi, & r > R \\ \frac{I\mu_0}{2\pi R}\frac{r}{R}\vec{e}_\phi, & r \leq R \end{cases} \quad (14.2)$$

We can solve the current from Ohms law to obtain the maximum field value at $r = R$ to be

$$|\vec{B}| = \frac{1}{2}\mu_0 R\sigma|\vec{E}| \quad (14.3)$$

Using electric conductivity of copper ($\sigma=59.59e6$ A/m²V) we obtain 0.0374 T.

We are interested in what happens when the voltage is applied sinusoidally with a frequency of 100 kHz. The harmonic case is unfortunately much more difficult to solve and no simple analytical expression exists. Therefore we need to solve the problem numerically.

Solution procedure

This tutorial will need the additional EDF, `magnetodynamics.xml`, for the WhitneyAVHarmonic Solver. The definitions for the relevant equation are not loaded into ElmerGUI by default. Hence, one needs to load these before starting the simulations.

```
File
  Definitions
    Append -> magnetodynamics.xml
```

The additional definitions should reside in the directory `edf-extra` within the distribution. Moving the desired `xml` files to the `edf`-directory enables automatic loading of the definitions at start-up. By inspecting the definitions in the Elmer Definitions File editor one may inspect that the new definitions were really appended.

The mesh is already defined in ElmerGrid format as file `wire.grd`. Load it from the tutorial directory where it resides.

```
File
  Open -> wire.grd
```

The ElmerGrid plug-in of ElmerGUI will read the mesh and create the mesh for ElmerGUI using the ElmerGrid plug-in. Note that the user could also create the mesh directly on the command line by

```
ElmerGrid 1 2 wire.grd
```

and then use the `Load Mesh` option in ElmerGUI to take the mesh into use. If the user wants to modify the default mesh that can be done by editing the file directly. The mesh has been constructed so that it can capture some boundary layer phenomena that we expect to take place on the surface of the wire. In principle the mesh could be even shorter since the results do not really depend on the axial direction.

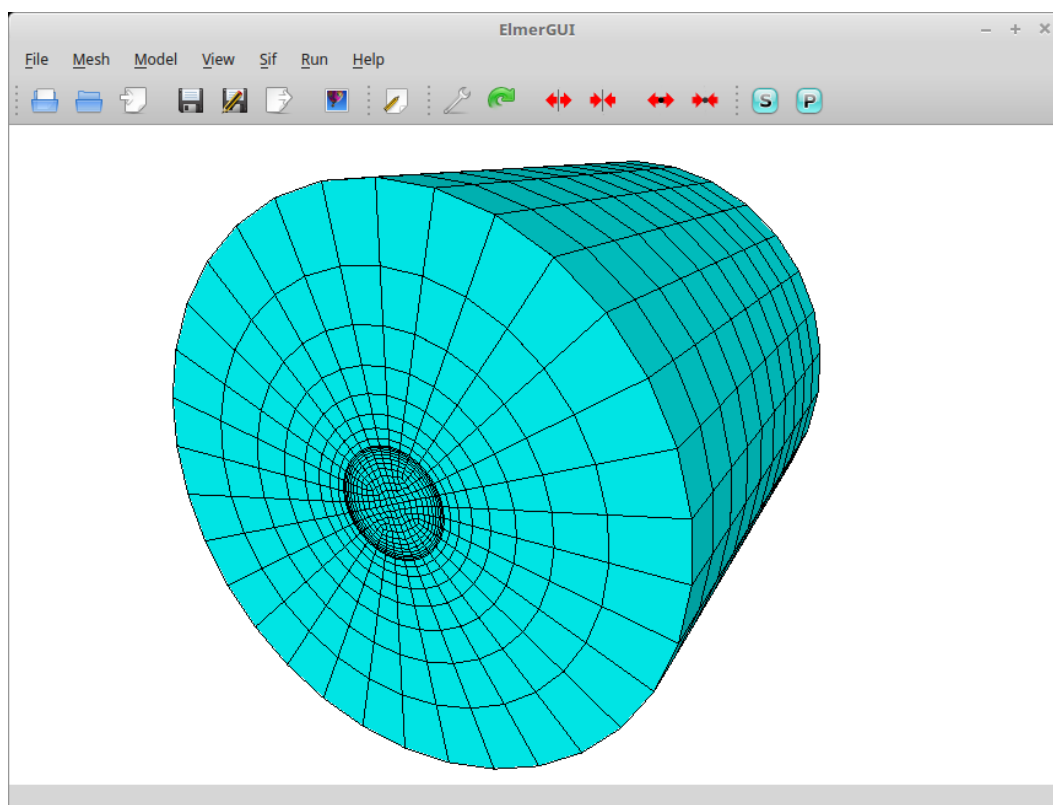


Figure 14.1: The mesh for the wire and surrounding air as seen in ElmerGUI

After we have the mesh we start to go through the `Model` menu from the top to bottom.

In the `Setup` we choose things related to the whole simulation such as file names, time stepping, constants etc. The steady-state simulation is carried out in 3-dimensional Cartesian coordinates. The initial size of the mesh is in millimeters, which is 1000 times too large. Hence we apply scaling in this menu, `Coordinate`

Scaling = 1.0e-3. Enter a single scaling number, and the scaling will be applied equally to all coordinates, whether in 2D or 3D.

Also we want to use a harmonic equation that requires the frequency to be set somewhere, in this case we will define the frequency in the Simulation section.

```
SetUp
Coordinate Scaling
  1.0e-3
Angular Frequency
  1.0e5
```

The permeability of vacuum is predefined in SI Units in the Constants section. To set other than the default value for it (in case using a different unit system), change the value in the Constants section.

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have the MgHarm solver (there exists also a steady-state/transient version of the solver as MgDyn), as well as the postprocessing solver MgDynPost.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we know that body 1 is the wire and body 2 is the surrounding air and we can apply the definitions directly. They will have the same set of solvers but different material properties.

The solver specific options may need some alternations. We choose a more suitable linear solver strategy for the AV equation.

For the postprocessing we add some fields to be computed and skip the computation of nodal fields since the elemental fields are often preferable for discontinuous fields. We also want to ensure that the postprocessing solver is run just before saving the results, after the primary solver has been computed.

In the following the correct equation are chosen and the suitable solver specific options are chosen:

```
Model
Equation
  Name = MgHarm
  Active = on
  Apply to Bodies = 1 2
  Edit Solver Settings
    Linear System
      Iterative = BiCGStabL
      Preconditioning = none
      BiCGStabl order = 4
  Name = MgDynPost
  Active = on
  Edit Solver Settings
    Solver Specific Options
      Calculate Magnetic Field Strength = on
      Calculate Joule Heating = on
      Skip Nodal Field = on
      Discontinuous Bodies = on
    General
      Execute Solver = before saving
Add
OK
```

The Material section includes all the material parameters. In this case we basically have two different materials – the copper wire and the surrounding air. We use the definitions from the small material database that comes with the code, and the material properties are in SI Units.

```
Model
Material
```



```

Material Library
  Copper
Apply to Bodies = 1
Add
New

```

```

Material
  Material Library
    Air (room temperature)
Apply to Bodies = 2
Add
OK

```

We need to set the boundary conditions both for the scalar potential associated to the nodal degrees of freedom, av , and to the vector potential associated to the edge degrees of freedom, $av \{e\}$. Both of these have both real and imaginary components. We set the scalar potential only for the conductors and the vector potential for all external boundaries.

```

Model
  BoundaryCondition
    Name = Ground
  MgHarm
    AV re = 0
    AV re e 1 = 0
    AV re e 2 = 0
    AV im = 0
    AV im e 1 = 0
    AV im e 2 = 0
  Add
  OK

```

For the other of the wire we have exactly the same BCs except that the scalar potential is set to 0.01 V to give the desired electric field of 1 V/m, since the length of the wire is 0.01 m.

```

Model
  BoundaryCondition
    Name = Voltage
  MgHarm
    AV re = 0.01
    AV re e 1 = 0
    AV re e 2 = 0
    AV im = 0
    AV im e 1 = 0
    AV im e 2 = 0
  Add
  OK

```

and finally for all other external BCs we set the boundary non-axial components of the vector potential to zero.

```

Model
  BoundaryCondition
    Name = AxialField
  MgHarm
    AV re e 1 = 0
    AV re e 2 = 0
    AV im e 1 = 0

```

```
AV im e 2 = 0
Add
OK
```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

```
Model
Set boundary properties
Choose one end of the wire -> set boundary condition to Ground
Choose the other end of the wire -> set boundary condition to Voltage
Choose all other external BCs -> set boundary condition to AxialField
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
Generate
Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case.

```
File
Save Project
```

After we have successfully saved the files we may start the solver

```
Run
Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. The equation is fully linear and hence only two iterations are needed – the second one just ensures that convergence of the non-linear level was really obtained. The convergence monitor also plots the postprocessing steps but they do not really converge as each field is computed just once.

When the solution has finished we may start the postprocessor to view some results.

```
Run
Start ParaView
```

Results

Here we present some results of the computations. The visualization is done using Paraview and the `vtu` format. For optimal visualization turn on the `Discontinuous Bodies` flag on both on the calculation of the fields and while saving the results in `vtu` format. These flags ensure that the solution is continuous within the bodies while maintaining jumps between bodies. No other formats in Elmer can currently support these features.

The resulting absolute values of the imaginary and real part of the magnetic flux density are depicted in Figure 14.2 and Figure 14.3. The imaginary part dominates in the amplitude (im 0.00522 T vs. re 0.00111 T). The corresponding vector field of the magnetic flux density is depicted in Figure 14.5.

Why are the results so far from the steady state solution? If you run the case again with 100 Hz you can see that the solution is very close to the static one as shown in Figure 14.4. The maximum magnetic flux density from the computations is 0.0366 T which is very close to the analytical solution of 0.0374 T. So the frequency really has an significant effect on the results!

You may study what happens when you increase the frequency further. At some point the mesh cannot capture the solution properly any more and the results become questionable.

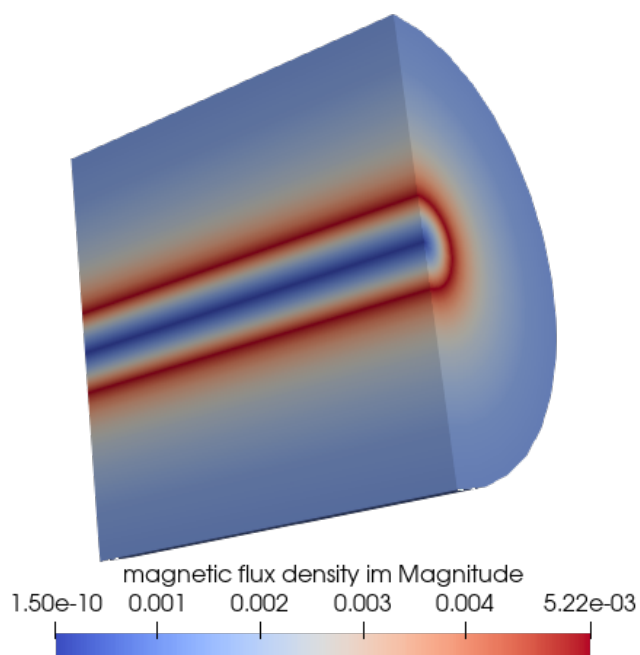


Figure 14.2: Imaginary part of the magnetic flux density at 100k Hz.

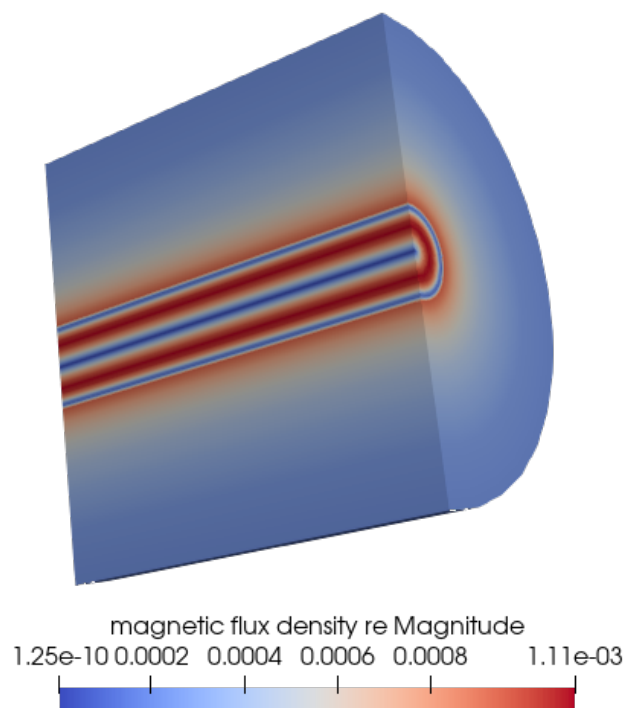


Figure 14.3: Real part of the magnetic flux density at 100k Hz.

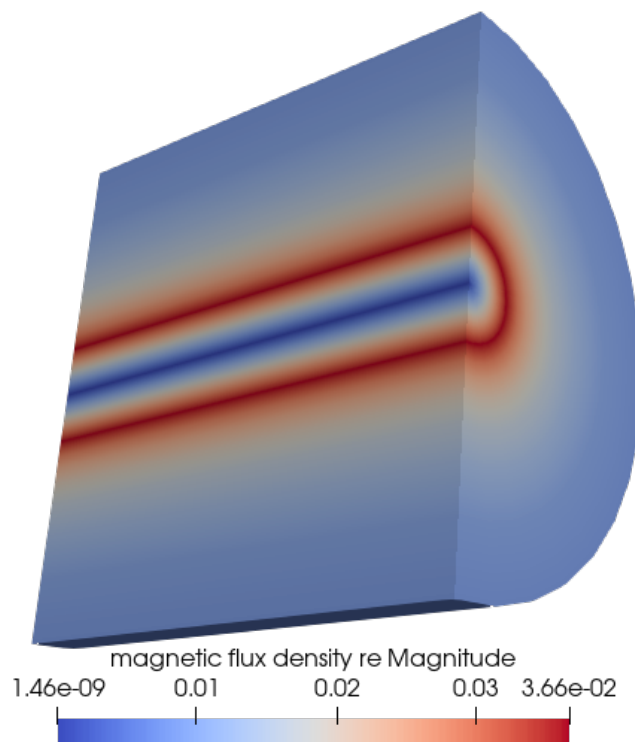


Figure 14.4: Real part of the magnetic flux density at 100 Hz.

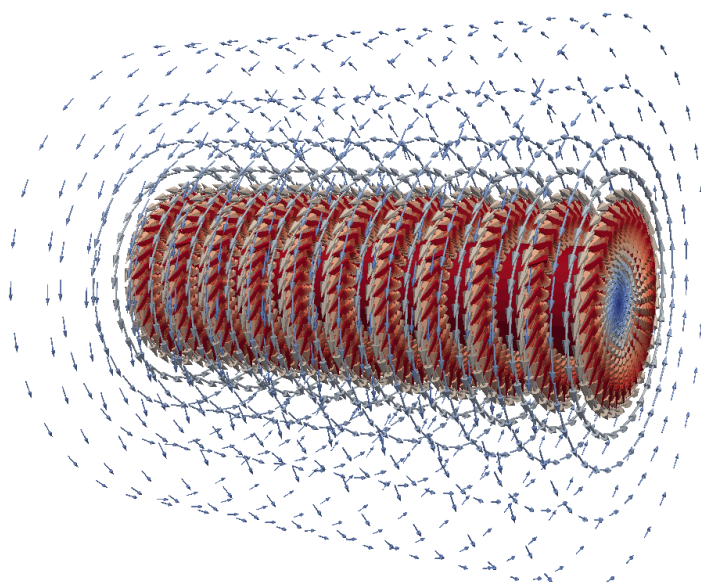


Figure 14.5: Imaginary part of the magnetic field strength plotted with vectors scaled by the field values.

Tutorial 15

Magnetostatics – Magnetic field resulting from a permanent magnet

Directory: HorseshoeGUI
Solvers: MagnetoDynamics2D
Tools: Gmsh, ElmerGUI
Dimensions: 2D, Steady-state
Author: Peter Råback

Case definition

This case roughly reproduces the case of a permanent magnet as demonstrated in the following link:
<http://www.strek.strefa.pl/students/meslec/lab06.pdf>

Consider a horseshoe-shaped permanent magnet. It consists of a ferromagnetic material but the two end sections are premagnetized in opposite directions. This results to a familiar magnetic field pattern. The horseshoe consists of three different regions and additionally there is the surrounding air. There is a circular outer boundary in order to conveniently allow for far field conditions. The material is assumed to have a constant relative permeability of 5000 and the magnetization is set to 750 kA/m.

Note that as this is a 2D case the resulting fields actually are those of an infinitely long horseshoe which of course does not make much sense in real life.

Meshing

The computational mesh is predefined in Gmsh format in file `horseshoe.msh`. If the user wants to modify the default mesh that must be done with Gmsh. The geometry of the file is given in file `horseshoe.geo`.

Solution procedure

The definitions for the relevant equation are not loaded into ElmerGUI by default. Hence, one needs to load these before starting the simulations.

```
File
  Definitions
    Append -> magnetodynamics2d.xml
```

The additional definitions should reside in the directory `edf-extra` within the distribution.

Optionally, one may copy the desired `xml` files to the `edf`-directory from the directory `edf-extra` which will enable automatic loading of the definitions every time ElmerGUI is started. By inspecting the definitions in the Elmer Definitions File editor one may verify that the new definitions were

really appended.

The mesh is already defined, load it from the `samples` directory where it resides.

File

Open -> `horseshoe.msh`

The ElmerGrid plug-in of ElmerGUI will read the mesh and convert it to a format understood by Elmer.

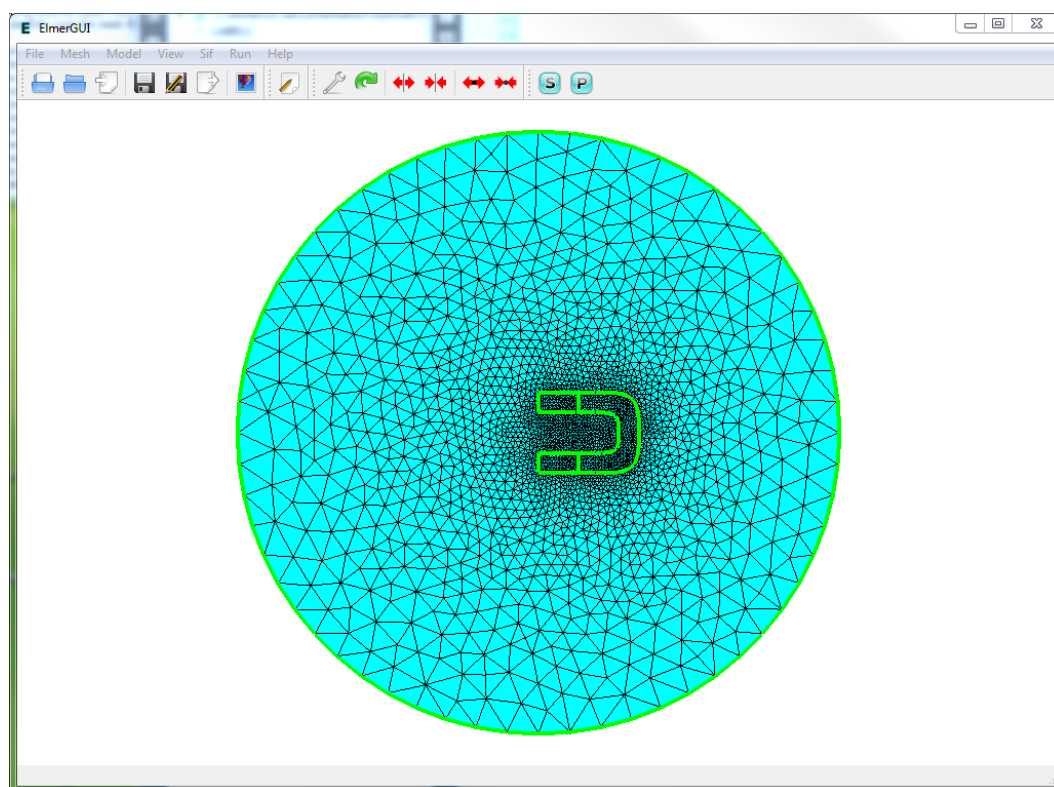


Figure 15.1: The mesh for the horseshoe and surrounding air as see in ElmerGUI

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. The steady-state simulation is carried out in 2-dimensional Cartesian coordinates. Nothing needs to be changed here. Currently the permeability of vacuum is not given in the ElmerGUI. To set other than the default value for it, the free text box can be used.

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have the MgDyn2D solver, as well as the postprocessing solver MgDyn2DPost.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case the equations need to be solved in all the bodies and hence clicking all from 1 to 4 here is most convenient. We give a higher priority to the actual solver so that the vector potential will be computed before the derived fields. In this case solver specific options should be OK but they could also be changed in this context.

Model

Equation

```
Name = MgDyn2D
Active = on
Priority = 1
Apply to Bodies = 1 2 3 4
```

```
Name = MgDyn2DPost
  Active = on
Add
OK
```

The Material section includes all the material parameters. In this case we basically have two different materials but the different magnetization must also be given as a material property. Hence we actually need to define four materials.

```
Model
  Material
    Name = Air
    MgDyn2d
      Relative Permeability = 1.0
    Add
    New

    Name = Iron
    MgDyn2d
      Relative Permeability = 5000.0
    Add
    New

    Name = IronPlus
    MgDyn2d
      Relative Permeability = 5000.0
      Magnetization 1 = Real 750.0e3
    Add
    New

    Name = IronMinus
    MgDyn2d
      Relative Permeability = 5000.0
      Magnetization 1 = Real -750.0e3
    Add
    OK
```

We may now assign the material properties by selecting with the mouse. This spares us of the need to know the indexes of each body.

```
Model
  Set body properties
    Choose air -> set Material to Air
    Choose curved part of horseshoe -> set Material to Iron
    Choose upper straight part of horseshoe -> set Material to IronPlus
    Choose lower straight part of horseshoe -> set Material to IronMinus
```

We have just one boundary condition i.e. the outer boundary for which we use the far field condition.

```
Model
  BoundaryCondition
    Name = Farfield
    MgDyn2D
      Infinity BC = True
    Add
    OK
```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

Model

Set boundary properties

Choose the 4 pieces of the outer sphere -> set boundary condition Farfield

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

Sif

Generate

Edit -> look how your command file came out

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case.

File

Save Project

After we have successfully saved the files we may start the solver

Run

Start solver

A convergence view automatically pops up showing relative changes of each iteration. The equation is fully linear and hence only two iterations are needed – the second one just ensures that convergence of the non-linear level was really obtained. The norm of the solution should be 0.3679.

When the solution has finished we may start the postprocessor to view some results.

Run

Start ParaView

Results

The resulting z-component of the vector potential is depicted in Figure 15.2. The corresponding postprocessed magnetic field intensity is depicted in Figure 15.3. Note that the derived fields is enforced to be continuous by default which is not optimal for visualization. For optimal results use Discontinuous Galerkin (DG) method for the postprocessing. Note that when using DG the postprocessing should be done with .vtu files and Paraview. The postprocessing tools of Elmer cannot deal with elementwise-fields.

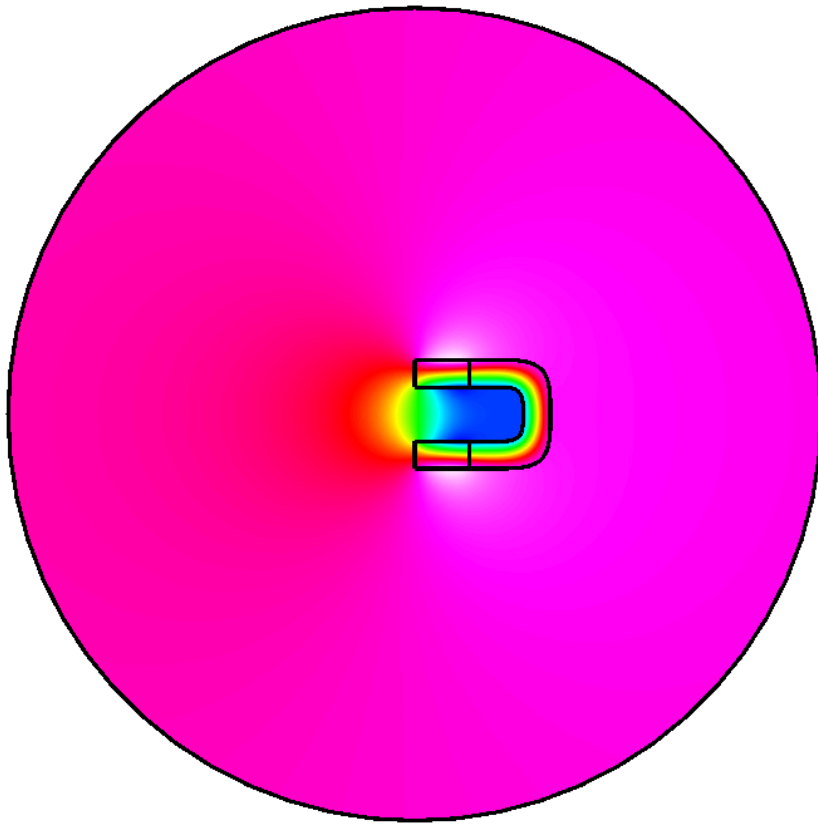


Figure 15.2: The vector potential of the magnetic field.

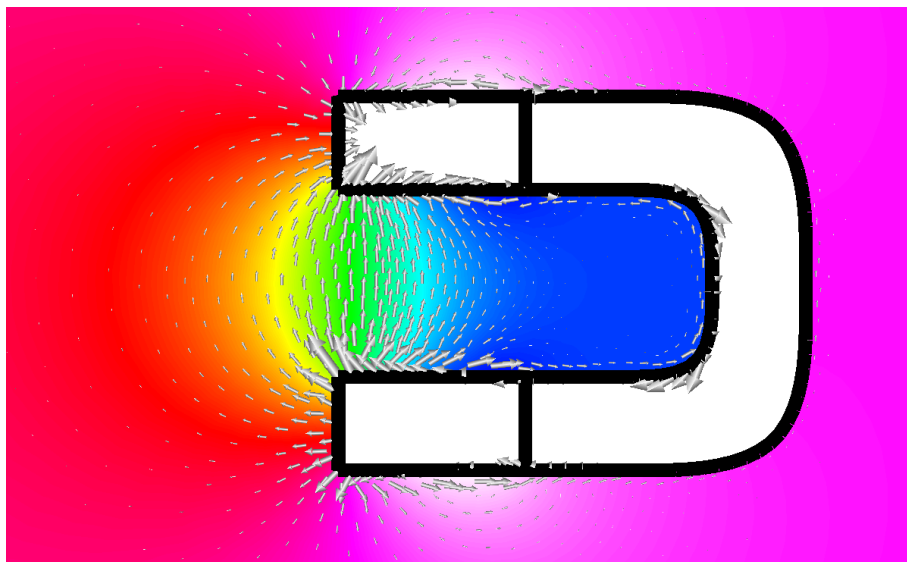


Figure 15.3: A closeup of the vector potential combined with the magnetic field intensity vectors. Note that the fields in the horseshoe itself have been masked away to demonstrate the well known field shape in the free space.

Tutorial 16

Harmonic magnetic field – 2D – Induction heating of a graphite crucible

Directory: InductionHeatingGUI

Solvers: MagnetoDynamics2DHarmonic

Tools: Gmsh, ElmerGUI

Dimensions: 2D, Axi-Symmetric

Author: Peter Råback

Case definition

At high temperatures the most practical method to heat up the crucible is by electromagnetic induction. The induction coil generates an alternating current that flows through the crucible. The Ohmic resistance encountered by this current dissipates energy, thereby directly heating the crucible via internal heat generation.

The tutorial case is a simple axi-symmetric crucible that could be used, for example, to grow silicon carbide (SiC) with physical vapour deposition. The crucible is made of dense graphite and isolated by porous graphite. At the bottom of the crucible there is some SiC powder. The physical properties of the material are given in Table 16.1. The dimensions of the induction heating crucible are given in Table 16.2.

We neglect the helicity of the coil and assume an average current density that may be computed easily when the area of the coil is known, $j_0 = nI/A$, where A is the coil area. Here we assume a current density of $1.0\text{e}6 \text{ A/m}^2$. The frequency of induction heating f is assumed to be 50 kHz.

The permeability of vacuum is $4\pi 10^{-7}$ if the other variables are in SI-units. Relative permeability is assumed to be one in all materials.

Solution procedure

The definitions for the relevant equation are not loaded into ElmerGUI by default. Hence, one needs to load these before starting the simulations.

File

Definitions

Table 16.1: Material parameters of the crucible

material	ϵ	κ [W/mk]	σ (1/ Ωm)
graphite	0.7	10.0	2.0E4
insulation	0.9	1.0	2.0E3
powder	0.5	25.0	1.0E4

Table 16.2: Dimensions of the crucible

body part	r_{inner}	r_{outer}	h_{inner}	h_{outer}
graphite	2.0	2.5	6.0	8.0
insulation	2.5	4.0	8.0	12.0
coil	5.0	5.5	8.0	8.0

```
Append -> magnetodynamics2d.xml
```

The additional definitions should reside in the directory `edf-extra` within the distribution. Moving the desired `xml` files to the `edf`-directory enables automatic loading of the definitions at start-up. By inspecting the definitions in the Elmer Definitions File editor one may inspect that the new definitions were really appended.

The mesh is already defined, load it from the `samples` directory were it resides.

```
File
```

```
Open -> crucible.msh
```

The ElmerGrid plug-in of ElmerGUI will read the mesh and convert it to a format understood by Elmer.

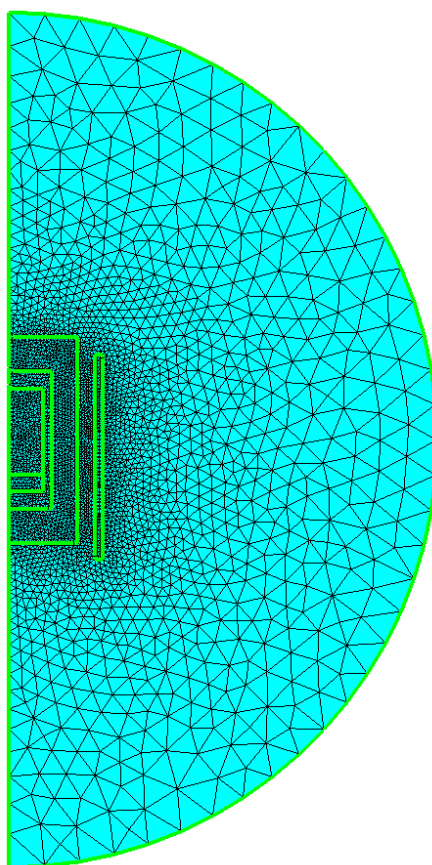


Figure 16.1: The mesh for the crucible and coil

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. Currently the permeability of vacuum is not given in the ElmerGUI. To set other than the default value for it, the free

text box can be used. The steady-state simulation is carried out in rotationally symmetric 2-dimensional coordinates which must be changed.

```
Model
  Setup
    Coordinate system -> Axi symmetric
```

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have the MgDyn2DHarmonic solver, as well as the postprocessing solver MgDyn2DPost.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case the equations need to be solved in all the bodies and hence clicking all from 1 to 6 here is most convenient. We give a higher priority to the actual solver so that the vector potential will be computed before the derived fields. In this case solver specific options should be OK but they could also be changed in this context.

```
Model
  Equation
  Solver -> MgDyn2DHarmonic
    Active = on
    Priority = 1
    Angular Frequency = 50.0e3
    Apply to Bodies = 1 2 3 4 5 6
  Solver -> MgDyn2DPost
    Active = on
    Edit Solver Settings
      Solver Specific Options
        Target Variable Complex = on
        Calculate Joule Heating = on
    Name = Induction
  Add
  OK
```

The Material section includes all the material parameters. In this case we basically have two different materials but the different magnetization must also be given as a material property. Hence we actually need to define four materials. The thermal properties are not needed at this stage as we are only solving for the induction. The internal losses of the coil are omitted and it is treated as a pure current source with material properties of air.

```
Model
  Material
    Name = Air
    MgDyn2dHarmonic
      Relative Permeability = 1.0
      Electric Conductivity = 0.0
    Add
    New

    Name = Graphite
    MgDyn2dHarmonic
      Relative Permeability = 1.0
      Electric Conductivity = 2.0e4
    Add
    New

    Name = Insulation
    MgDyn2dHarmonic
```

```

    Relative Permeability = 1.0
    Electric Conductivity = 2.0e3
Add
New

```

```

Name = Powder
MgDyn2dHarmonic
    Relative Permeability = 1.0
    Electric Conductivity = 1.0e4
Add
New

```

We may now assign the material properties by selecting with the mouse. This spares us of the need to know the indexes of each body.

```

Model
Set body properties
    Choose external air -> set Material to Air
    Choose coil -> set Material to Air
    Choose insulation (outermost body) -> set Material to Insulation
    Choose graphite (actual crucible) -> set Material to Graphite
    Choose powder at the bottom of crucible -> set Material to Powder
    Choose internal air above the powder-> set Material to Air

```

We need to provide a current source in order for the equation to have a non-trivial solution.

```

Model
BodyForce
    Name = CurrentSource
    MgDyn2DHarmonic
        Current Density = 2.5e5
Add
OK

```

This must be also joined with the coil

```

Model
Set body properties
    Choose coil -> set Body force to CurrentSource

```

We have just one boundary condition i.e. the outer boundary for which we use the far field condition.

```

Model
BoundaryCondition
    Name = Farfield
    MgDyn2DHarmonic
        Infinity BC = True
Add
OK

```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

```

Model
Set boundary properties
    Choose the 2 pieces of the exterior -> set boundary condition Far field

```

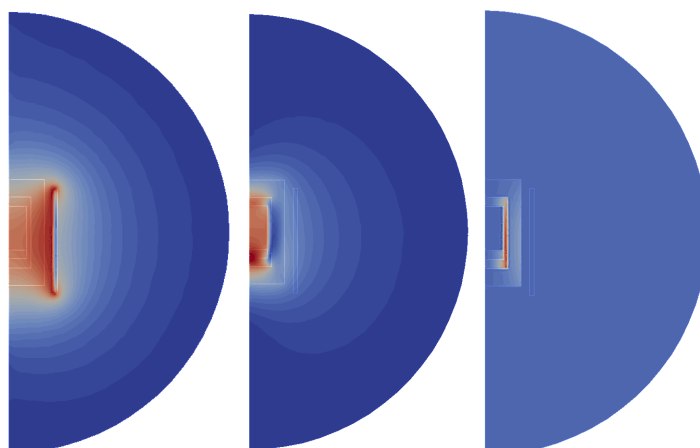


Figure 16.2: Induction heating of a simple crucible. a) in-phase component of the magnetic field intensity b) out-of-phase component of the magnetic field intensity c) Joule losses in the conductors

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
  Generate
  Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case.

```
File
  Save Project
```

After we have successfully saved the files we may start the solver

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. The equation is fully linear and hence only two iterations are needed – the second one just ensures that convergence of the non-linear level was really obtained. The norm of the solution should be 0.3679.

When the solution has finished we may start the postprocessor to view some results.

```
Run
  Start ParaView
```

With the given computational mesh the problem is solved in a few seconds. With the 6 542 linear triangles the heating efficiency is estimated to be 18.9 W. The corresponding results are shown in Fig. 16.2.

It can be noted that would the estimated heating efficiency be different from the known one the user may give the postprocessing solver the `Desired Heating Power` in order to scale the potential of the solution. The solution, on the other hand, may be used as a source term in heat equation, for example.

Tutorial 17

Helmholtz – 2D – Acoustic Waves – Air in a Cavity

Directory: AcousticWaves

Solvers: Helmholtz

Tools: ElmerGUI

Dimensions: 2D, Steady State

Author: original author not known

Introduction

Elmer provides two alternative ways of conducting acoustic analyses in the frequency domain. Firstly, one may simply use the Helmholtz equation which is based on the assumption of lossless flow, i.e. the effects of viscosity and heat conduction are assumed to be negligible. More refined analyses where these effects are taken into account may be carried out by using the specific solver for the set of time-harmonic dissipative acoustic equations. The aim of this tutorial is to demonstrate the usage of the solver for the basic Helmholtz equation, which is frequently taken as the starting point in acoustic analyses.

Case definition

In this problem the fluctuations of the pressure in an air-filled cavity shown in Figure 31.1 are considered. The cavity is connected with the surrounding air by an open narrow pipe. The pressure fluctuations are generated by a vibrating membrane on the boundary Γ_S with the frequency of the motion being $f = 100$ Hz. The remaining parts of the boundary are assumed to be rigid walls. In addition, the effects of gravity are assumed to be negligible.

Suitable boundary conditions in terms of the pressure must be given. On the rigid walls the pressure flux is prescribed to vanish which corresponds to the assumption that there is no velocity in the direction normal to the boundary. At the open end Γ_0 the impedance boundary condition suitable for forward travelling plane waves is given by setting $Z = -c$ with c being the sound speed. We assume that $c = 343$ (m/s). Finally, the wave source is given by defining a non-vanishing pressure flux on

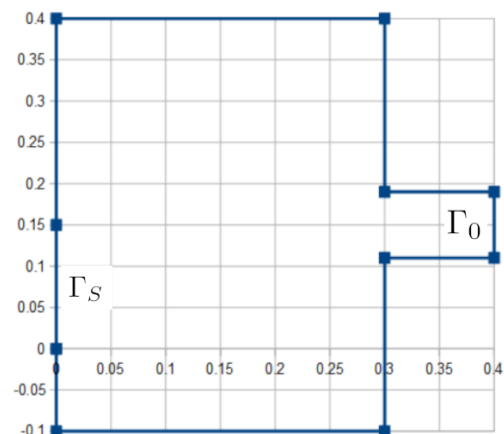


Figure 17.1: Geometry

the corresponding part of the boundary. We take simply $\nabla P \cdot \vec{n} = 1$ where P is the (complex) amplitude of the pressure and \vec{n} is the outward unit normal to the boundary.

ElmerGUI Equation Menu

We will be using the Helmholtz Solver, which is one of the default, pre-loaded GUI definitions, so it does not need to be manually activated. For reference, the `helmholtz.xml` definition file is, by default, located in Linux in:

```
$ELMER_HOME/share/ElmerGUI/edf
```

and in Windows, located in:

```
C:/Program Files/Elmer 9.0-Release/share/ElmerGUI/edf
```

Solution procedure

The mesh is given in ElmerGrid format in file `domain.grd`, load this file.

File

```
Open -> domain
```

You should obtain your mesh and may check Model Summary... that it consists of 10050 surface elements. Your mesh should look like as shown in figure 31.5

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc.

The simulation is carried out in 2-dimensional Cartesian coordinates.

Model

Setup

```
Simulation Type = Steady state
```

```
Steady state max. iter = 1
```

Apply

In the equation section we choose the relevant equations and parameters related to their solution.

In this case we'll have one equation (named "Helmholtz").

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and therefore its easier to assign the Equation and Material to it directly.

For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1,..) or direct Umfpack solver, for example.

Model

Equation

```
Name = Helmholtz
```

```
Apply to Bodies = 1
```

```
Helmholtz Equation
```

```
Active = on
```

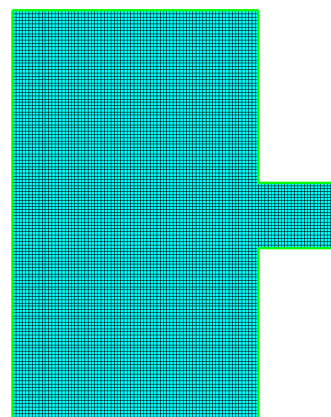


Figure 17.2: Mesh.


```

Angular Frequency = 628.3
Convection Velocity 1 = 0.0
Convection Velocity 2 = 0.0
Add
OK

```

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such as conductivities and viscosity.

```

Model
Material
Apply to Bodies = 1
General
Density = 1.224
Helmholtz
Damping coefficient = 0.0
Sound speed = 343
Add
OK

```

A Body Force represents the right-hand-side of a equation. It is generally not a required field for a body, and is not needed for this example.

Initial conditions should be given to transient cases, and probably are not needed for steady state solutions, and is not needed for this example.

Only one boundary condition may be applied to each boundary and therefore all the different physical BCs for a boundary should be grouped together.

The conditions may be assigned to boundaries in the Boundary condition menu, or by clicking on each boundary with the mouse. Here we use the first approach since there are only three boundaries.

```

Model
BoundaryCondition
Name = Constraint1
Boundary 1 is checked
Helmholtz Equation
Flux conditions
Real part of the flux = 1
Imag part of the flux = 0
Add
New

Name = Constraint2
Boundary 2 is checked
Helmholtz Equation
Flux conditions
Real part of the flux = 0
Imag part of the flux = 0
Add
New

Name = Constraint3
Boundary 3 is checked
Helmholtz Equation
Flux conditions
Real part of the impedance = -343

```

```
    Imag part of the impedance = 0
  Add
  OK
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
  Generate
  Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The ElmerGUI project includes all the files needed to restart the case.

```
File
  Save Project
```

After we have successfully saved the files we may start the solver.

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. When there are some results to view we may start the postprocessor also.

```
Run
  Start ParaView
```

Results

The difference of the maximum of (0.32) and the minimum (0.21) value of the pressure is found to be $\Delta p \approx 0.11$

You may inspect the results with Paraview or with ElmerVTK.

In Figure 17.3 the obtained pressure wave magnitude is presented. In Figure 17.4 the obtained pressure wave in the x-direction and y-direction is presented.

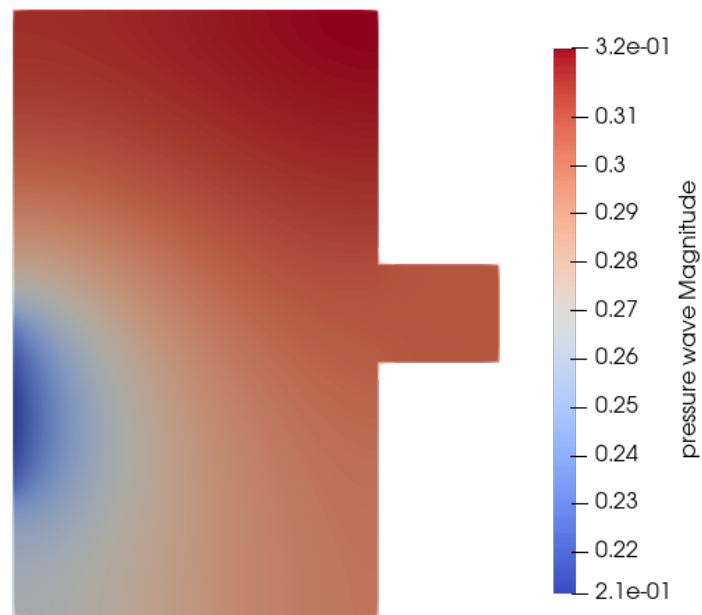


Figure 17.3: Pressure wave magnitude

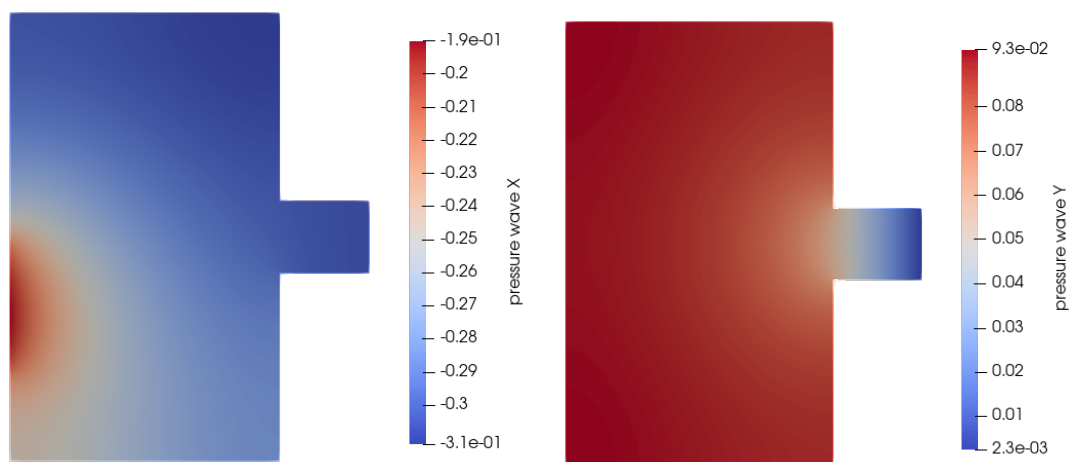


Figure 17.4: Pressure wave, x-direction, and y-direction

Extra task:

If you have time you may try to solve the case with different parameters.

Tutorial 18

VectorHelmholtz – model wave propagation in bent waveguide

Directory: WaveguideGUI

Solvers: VectorHelmholtz

Tools: ElmerGUI

Dimensions: 3D, Steady-state

Author: Juhani Kataja

Case definition

In this tutorial we model propagation of a guided wave through a bend in a rectangular waveguide. Let us choose the primary wave to be a TE_{10} mode at $f = 2.5$ GHz propagating in the positive z axis direction (time-harmonic convention $e^{-i\omega t}$).

The electric field of the primary wave is given by

$$\vec{E}_p = \vec{u}_y \frac{ik}{k_c} \sin(k_c x) e^{i\beta z},$$

where $k_c = \pi/a$, $k = \omega\sqrt{\varepsilon_0\mu_0}$ and $\beta = \sqrt{k^2 - k_c^2}$. The dimensions of the waveguide's cross section are $a = 10$ cm and $b = 5$ cm, the bend is a piece of circular torus with outer radius of 12 cm and inner radius of 2 cm. The waveguide geometry is depicted in Fig. 18.1.

The input port is that part of the boundary which is normal to z -axis, the output port that which is normal to x -axis and the rest of the boundary is a PEC boundary.

The PEC boundary condition is given by $\vec{n} \times \vec{E} = 0$, the input boundary condition by

$$\vec{n} \times \nabla \times \vec{E} - i\beta\vec{n} \times (\vec{n} \times \vec{E}) = i2\beta\vec{n} \times (\vec{n} \times \vec{E}_p),$$

and the output boundary condition by

$$\vec{n} \times \nabla \times \vec{E} - i\beta\vec{n} \times (\vec{n} \times \vec{E}) = 0.$$

Here \vec{n} is the exterior normal of the domain.

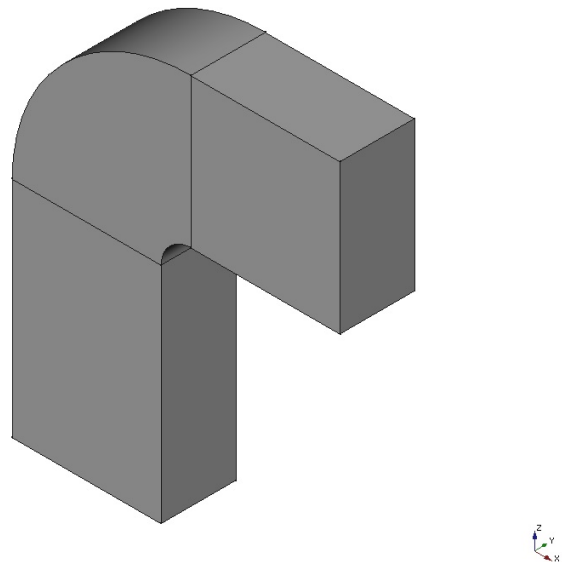


Figure 18.1: Geometry of the waveguide

Load ElmerGUI Equation Menu

The GUI definitions of VectorHelmholtz solver utilized in this tutorial are located in the `edf-extra` folder and thus need to be manually activated in the ElmerGUI.

Note that the extra definition needs to be loaded first, before any other steps in building any tutorial. Once the extra definition has been loaded, and the ElmerGUI project has been saved, then the instructions for the extra definition will be stored in the project, and you won't have to load the extra definition again.

```
File
  Definitions
    Append -> vectorhelmholtz.xml
  Close
```

The `vectorhelmholtz.xml` definition file is, by default, located in Linux in:

```
$ELMER_HOME/share/ElmerGUI/edf-extra
```

and in Windows, located in:

```
C:/Program Files/Elmer 9.0-Release/share/ElmerGUI/edf-extra
```

We will also be using the Result Output Solver, which is one of the default, pre-loaded GUI definitions, so it does not need to be manually activated. For reference, the `resultoutput.xml` definition file is, by default, located in Linux in:

```
$ELMER_HOME/share/ElmerGUI/edf
```

and in Windows, located in:

```
C:/Program Files/Elmer 9.0-Release/share/ElmerGUI/edf
```

After reading over several of the preceding tutorials that wrote results to the `case.vtu` file, and none of those tutorial used Result Output Solver, you may be wondering why we now all of a sudden need to load a special solver to output results?

Actually, the Result Output Solver has been loaded in each of the preceding tutorials, but it was done in the background, without notifying the user. All one needs to do, is to enter a file name into the Post file field in the Setup menu, as follows.

```
Model
  Setup
    Post file -> case.vtu
  Apply
```

This will cause Result Output Solver to be loaded in the background and will run after saving the simulation. To get back to your question, why add an equation entry for Result Output, the answer will be that we want to demonstrate how to use some of the advanced features of the Result Output Solver. Please take a look at the Elmer Models Manual and review the Result Output Solver. One of the features that we will use, is to ask for the `.vtu` file to be output in ASCII, instead of in binary.

Another feature of Result Output Solver is the ability to save the results in formats other than `.vtu`, including `vtk`, `Dx`, `Gmsh`, and `GiD` formats.

One last point, either enter a `'case.vtu'` name into the Post file field of Setup, or add Result Output Solver to an Elmer project. Don't add both, it won't hurt anything, but it will output two `.vtu` files, one will be valid and the other will be empty.

ElmerGUI Solution Procedure

Load geometry:

File

Open -> waveguide_bend.step

As we are modelling a wave phenomenon at 2.5GHz the maximum H (maximum element size) in the mesh must be around $\frac{1}{10\lambda}$, where $\lambda \approx 8.3$, so set Max H = 0.012. We mentioned the waveguide cross section is 10 cm by 5 cm, which equals 0.100 meters by 0.050 meters. Taking the smaller dimension and dividing by Max H, gives $0.050/0.012 =$ approximately four elements across the smaller dimension.

Within ElmerGUI, since the geometry input file is in Step format, Nglib from Netgen is used to mesh the model into linear tetrahedral elements. Nglib by default will generate a mesh where at least two elements will span the smaller dimension of the model. In this case, setting Max H = 0.012 to obtain about four elements across the smaller dimension is a necessary step.

Note that one could use Elmergrid to increase the order of the tet mesh generated by nglib from linear to quadratic, (from a command prompt, enter `elmergrid 2 2 -increase`) although in this case we keep the linear mesh as generated by nglib.

Mesh

Configure..

Max H: 0.012

Apply

Mesh

Remesh

Add the VectorHelmholtz and VectorHelmholtzCalcFields solver modules:

Model

Equation -> Add..

Vector Helmholtz Post Process

Active = on

Priority 4

Result Output

Active = on

Priority 2

Edit solver settings

Solver specific options

Uncheck the box 'Binary Output'

Apply

Vector Helmholtz Equation

Active = on

Apply to Bodies: Body 1

Angular Frequency = $2\pi \cdot 2.5e9$

Priority 5

Free text input

$a = 10e-2$

$b = 5e-2$

$c_0 = 1/\sqrt{8.854e-12 \cdot 4\pi \cdot 10^{-7}}$

$\omega = 2\pi \cdot 2.5e9$

$k_0 = \omega/c_0$

$k_c = \pi/a$

$\beta_0 = \sqrt{k_0^2 - k_c^2}$

Edit Solver Settings

Linear system

```

    Iterative = BiCGStabl
    Preconditioning = vanka
    BiCGStabl order = 6
    Convergence tol = 1e-6
    Apply
  Add
OK

```

Here in the Free text input part we defined some constants that will make definition of the rest of the model easier.

Then add boundary PEC conditions:

```

Model
Boundary Condition -> Add..
Name = PEC
VectorHelmholtz Equation
  E re {e} = 0
  E im {e} = 0
  Apply to boundaries = 2...13
  Add
OK

```

Note that because the mesh is tetrahedral, the $E_{re/im} \{f\}$ Dirichlet condition is unnecessary. The hexahedral and pyramidal Piola transformed elements have DOFs on element faces rendering the $\{f\}$ Dirichlet conditions meaningful.

Add the input port boundary condition:

```

Model
Boundary Condition -> Add..
Name = Inport
Apply to boundaries = 14
VectorHelmholtz Equation
  Magnetic Boundary Load 2 [enter]
  Variable Coordinate 1
  Real MATC "-2*beta0*k0/kc*sin(kc*(tx+a/2))"
  Close
  Electric Robin Coefficient im = $ beta0
  Add
OK

```

Next add the output port boundary

```

Model
Boundary Condition -> Add..
Name = Outport
Apply to boundaries = 1
VectorHelmholtz Equation
  Electric Robin Coefficient im = $ beta0
  Add
OK

```

Assign material to the body

```

Model
Material -> Add..
Apply to bodies: Body 1

```

```
Relative Permittivity = 1
Add
OK
```

Now navigate to choose what results are to be saved:

```
Model
Equation -> Equation 1
Vector Helmholtz Post Process
  Edit Solver Settings
    Solver specific options
      Calculate Electric Field = on
      Calculate Magnetic Field Strength = on
      Calculate Poynting Vector = on
      Calculate Energy Functional = on
    Apply
  Update
  OK
```

Save the project

```
File
Save project -> [choose location]
```

And solve:

```
Run
Start Solver
```

The resulting fields should now appear in case0001.vtu file ready for further post processing.

Results

We are interested in the Energy Functional Value in the solver log. It should read (prependded with “VectorHelmholtzSolver:”)

```
Energy Functional value: -11284.937620324963          453999.53923919413
```

The first number is the real part and second the imaginary part. Denoting this with $l(E)$ it holds that in this case

$$l(E) = \frac{i\beta k_0^2 ab}{\mu k_c^2} (1 + \rho),$$

where ρ is the reflection coefficient of electric field. Thus $\rho \approx -0.022 + 0.024i$, which translates to roughly 29.7 dB return loss.

In Fig. 18.2 the Poynting vector and the real part of the electric field of the solution field is shown.

Extra task:

Try outputting results using the .vtk format, and loading the .vtk file using Paraview. Also try saving the results in binary as well as in ASCII, and compare the resulting .vtu files using a text editor.

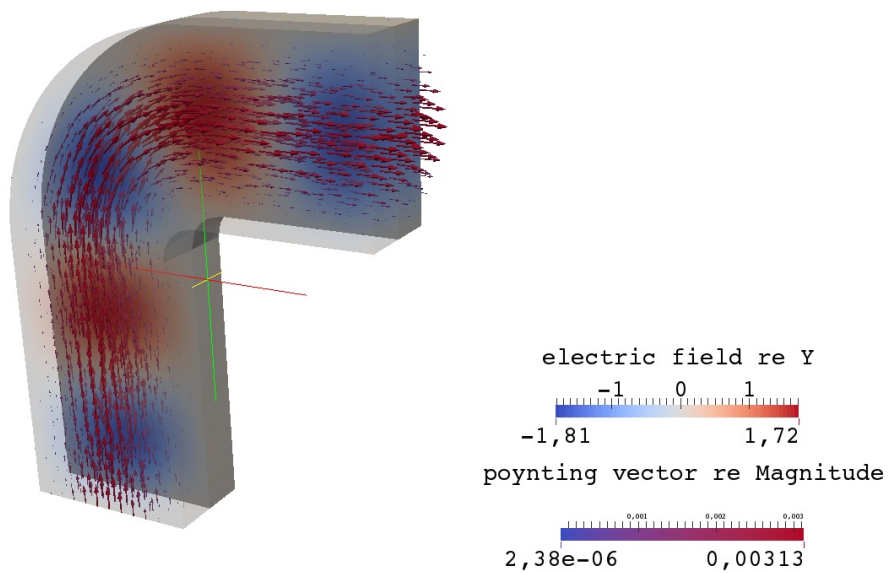


Figure 18.2: The resulting field solution: The y -component of the real part of the electric field and the real part of Poynting's vector

Tutorial 19

Navier-Stokes equation – Laminar incompressible flow passing a step

Directory: FlowStepGUI

Solvers: FlowSolve

Tools: ElmerGUI

Dimensions: 2D, Steady-state

Author: Peter Råback

Case definition

This tutorial represents the canonical step flow of viscous fluid. A fluid, flowing past a step (see figure 19.1), has the density 1 kg/m and viscosity 0.01 kg/ms. The velocity profile at the inlet is parabolic with a mean velocity $\langle v_x \rangle = 1.0$ m/s and $v_y = 0.0$ m/s. At the outlet only the vertical component is defined, $v_y = 0.0$ m/s. At all other walls the no-slip boundary condition, $\vec{v} = 0$, is applied. Thus the Reynolds number for the case is around 100.

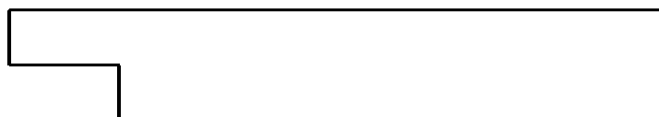


Figure 19.1: Geometry of the step flow problem

Mathematically the problem to be solved is

$$\begin{cases} -\nabla \cdot (2\mu\bar{\varepsilon}) + \rho\vec{u} \cdot \nabla\vec{u} + \nabla p = 0 & \text{in } \Omega \\ \nabla \cdot \vec{u} = 0 & \text{in } \Omega \end{cases} \quad (19.1)$$

with the boundary conditions

$$\begin{cases} u_x = 1 & \text{on } \Gamma_{inlet} \\ u_x = 0 & \text{on } \Gamma_{no-slip} \\ u_y = 0 & \text{on } \Gamma_{inlet} \cup \Gamma_{outlet} \cup \Gamma_{no-slip} \end{cases} \quad (19.2)$$

where μ is the viscosity, $\bar{\varepsilon}$ is the strain tensor, ρ is the density, \vec{u} is the velocity and p is the pressure. It is assumed that the density and viscosity are constants.

Solution procedure

The mesh is given in ElmerGrid format in file `step.grd`, load this file.

File

Open -> step.grd

You should obtain your mesh and may check that it consists of 9696 nodes and of 9442 bilinear elements.

Model

Summary...

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation. The steady-state simulation is carried out in 2-dimensional Cartesian coordinates, which are also the defaults.

Model

Setup

Simulation Type = Steady state
Coordinate system = Cartesian

In the equation section we choose the relevant equations and parameters related to their solution. In this case the only the Navier-Stokes equation is needed.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and therefore its easier to assign the Equation and Material to it directly. One could also edit the solver setting in order to try different strategies for solving the non-linear or linear system. Initially the Navier-Stokes solver uses the more robust Picard iteration which is changed to Newton iteration after few initial steps. For the given viscosity the default values are ok, but may need tuning when going into higher Reynolds numbers.

Model

Equation

Name = Navier-Stokes
Apply to Bodies = Body 1
Navier-Stokes
Active = on
Edit Solver Setting
Nonlinear System
Max. iterations = 20
Newton after iterations = 3

Add

OK

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the density. Other properties assume a physical law, such as viscosity.

Model

Material

Name = Ideal
General
Density = 1.0
Navier-Stokes
Viscosity = 0.01
Apply to Bodies = Body 1
Add
OK

The current case does not have any body forces. Convergence should also be obtained using the default initial condition which sets all field values to zero. Hence no setting for initial condition are needed.

Only one boundary condition may be applied to each boundary and therefore all the different physical BCs for a boundary should be grouped together. In this case the Temperature and Velocity. The side walls are assumed to be adiabatic.

The parabolic inlet-profile is achieved using the MATC environment. To be able to edit the content of the inlet profile click Enter to open an edit box for the Velocity 1. The given expression will be interpreted at run-time so that $v_x = 6(y - 1)(2 - y)$. As $y \in [1, 2]$ thereby creating a parabolic velocity profile with a mean velocity of unity.

```

Model
  BoundaryCondition
    Name = Inlet
    Navier-Stokes
      Velocity 1 = Variable Coordinate 2; Real MATC "6*(tx-1)*(2-tx)"
      Velocity 2 = 0.0
    Add
  New

  Name = Outlet
  Navier-Stokes
    Velocity 2 = 0.0
  Add
  New

  Name = Walls
  Navier-Stokes
    Noslip wall BC = on
  Add
  OK

```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

```

Model
  Set boundary properties
    Choose Inlet -> set boundary condition Inlet
    Choose Outlet -> set boundary condition Outlet
    Choose Walls -> set boundary condition Walls

```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```

Sif
  Generate
  Edit -> look how your command file came out

```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case. Create a suitable directory for the case if needed.

```

File
  Save Project

```

After we have successfully saved the files we may start the solver

```

Run
  Start solver

```

A convergence view automatically pops up showing relative changes of each iteration. The problem should converge in about ten iterations to a norm of 0.4347 visible on the output.

When there are some results to view we may start the postprocessor also

```

Run
  Start ParaView

```

Results

The results may be viewed using the postprocessor as shown in Figure 19.2 and 19.3. One may also register specific values, for example the pressure difference is 0.388 Pa, the minimum and maximum lateral velocities are -0.1666 m/s and 1.5 m/s, respectively. One special result of interest is the point, on the x-axis, at which the direction of the flow changes. In this case its position is about 5.0 m after the step.

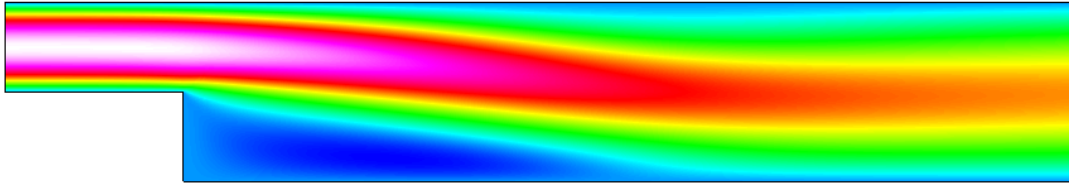


Figure 19.2: Absolute value of the velocity field



Figure 19.3: Pressure field

Extra task: Decreasing the viscosity

Try what happens if the viscosity is further decreased by a factor 10. Convergence may be difficult to obtain. Some tricks that may be tested include

- Introducing a relaxation factor (typically in the range 0.5–0.7)
- Increasing number of non-linear iterations
- Favoring Picard iteration over Newton
- Increasing mesh density (and length of domain)

Don't be worried if you fail to find convergence. This task will mainly act as a motivator in using turbulence models for higher Reynolds numbers.

Remember to re-perform the following phases in order to get the updated results

```
Sif
  Generate
File
  Save Project
Run
  Start solver
```

You may just reload the results in the postprocessor rather than closing and opening the program.

Tutorial 20

Navier-Stokes equation – 2D – Incompressible – Driven Cavity

Directory: DrivenCavity

Solvers: FlowSolve

Tools: ElmerGrid, ElmerGUI

Dimensions: 2D, Steady-state

Author: Evangelos Voyiatzis

Introduction

This simple tutorial illustrates the application of Elmer to computational fluid dynamics problems. The computational mesh is generated using the ElmerGrid utility while the model definition and its solution are carried out via ElmerGUI. The results are visualized with ParaView.

Case definition

The geometrical domain, which is shown in Figure 20.1, is a square with a side length of 1 m. The square is filled with an ideal fluid with a density of 1 kg/m^3 and a viscosity of 0.01 kg/m s . The boundary Γ_{A-B} is moving with a constant speed of 4 m/s in the x Cartesian direction while the no slip boundary condition is applied to the rest of the boundaries. The flow is laminar with a Reynolds number of 400. This system has been studied in detail by Ghia, Ghia and Shin in *J. Comp. Phys.* 48 (1982) 387-411 and it is also part of the NASA repository <https://www.grc.nasa.gov/WWW/wind/valid/cavity/cavity.html>.

The study by Ghia, et al, is run with a range of Reynolds numbers, from 100 up to 10,000. We will examine the case of Reynolds number equal to 400. Using SI units, the driven lid moves at 4 m/s, the viscosity equals 0.01 Pa-s, and the density equals 1.0 kg/m^3 . The values are chosen to provide the desired Reynolds numbers from the Ghia article simply by changing the driven lid velocity. For example, with the lid moving at 4 m/s we have a Reynolds number of 400, if the lid moves at 1 m/s we will have a Reynolds number of 100, and so forth.

The study by Ghia, et al, presents numerical results in two tables. The tabular results are the u velocity (in the x -direction) along a vertical line through the geometrical center and the v velocity (in the y -direction) along a horizontal line through the geometrical center. The two lines are shown in Figure 20.1. A comparison of the tabulated results from Ghia against the results generated by Elmer will be presented.

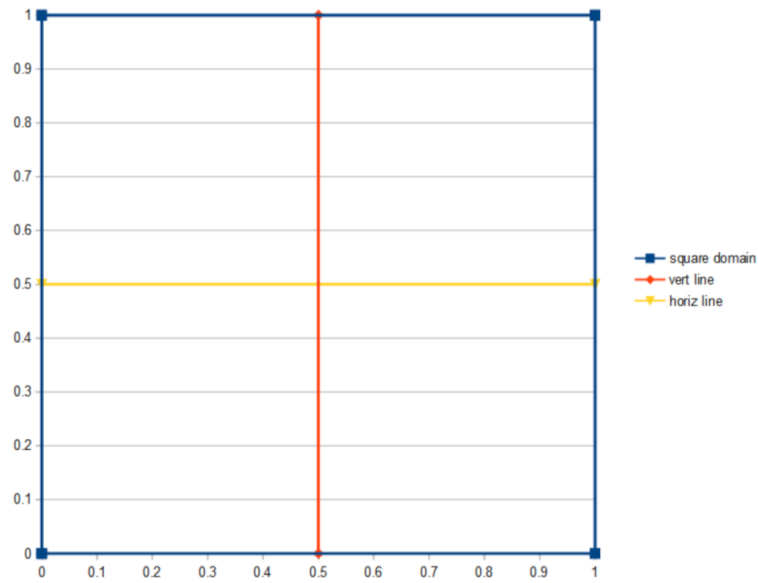


Figure 20.1: Geometry of the driven cavity problem

Mathematically the problem to be solved is

$$\begin{cases} -\nabla \cdot (2\mu\bar{\varepsilon}) + \rho\vec{u} \cdot \nabla\vec{u} + \nabla p = 0 & \text{in } \Omega \\ \nabla \cdot \vec{u} = 0 & \text{in } \Omega \end{cases} \quad (20.1)$$

with the boundary conditions

$$\begin{cases} u_x = 4 & \text{on } \Gamma_{A-B} \\ u_x = 0 & \text{on } \Gamma_{B-C} \cup \Gamma_{C-D} \cup \Gamma_{D-A} \\ u_y = 0 & \text{on } \Gamma \end{cases} \quad (20.2)$$

where μ is the viscosity, $\bar{\varepsilon}$ is the strain tensor, ρ is the density, \vec{u} is the velocity and p is the pressure. It is assumed that the density and viscosity are constants.

Solution procedure

The first step is the generation of a uniform computational mesh with 128 elements in each of the x and the y coordinate directions. This is achieved by the following input file for ElmerGrid which is stored in the "DrivenCavity.grd" file, which may be found in the tutorials-GUI-files/DrivenCavity folder:

```
*** ElmerGrid input file for structured grid generation ***
Version = 210903
Coordinate System = Cartesian 2D
Subcell Divisions in 2D = 1 1
Subcell Limits 1 = 0 1
Subcell Limits 2 = 0 1
Material Structure in 2D
  1
End
Materials Interval = 1 1
Boundary Definitions
! type out int double of the boundaries
  1   -1   1   1
  2   -2   1   1
  3   -3   1   1
  4   -4   1   1
End
Element Degree = 2
Surface Elements = 16384
```

Then start ElmerGUI and create a new project, as follows:

Run

New Project...

Start at the top and select the project directory as shown in Figure 31.3. Then select the Geometry input file, DrivenCavity.grd. Lastly, the SaveLine Equation Definition File is needed for the tutorial, so select SaveLine in the right hand box and add it to the left hand box. Click on OK to accept the changes.

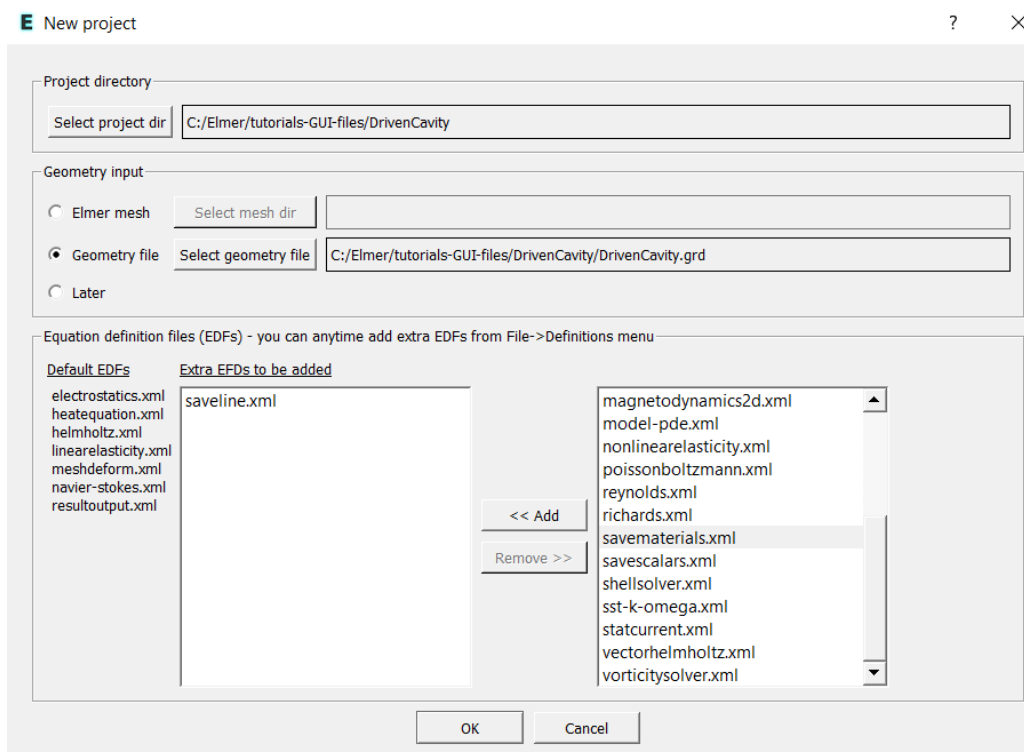


Figure 20.2: New Project

ElmerGUI Equation Definition Files

The New Project screen automatically locates and lists all of the available EDFs. The left side of the screen lists the Default EDFs, which will be loaded into each ElmerGUI project. The right hand box lists all of the extra EDFs, that are not normally loaded, allowing the option to add individual extra EDFs to a new project.

This tutorial will use the Default EDF, `navier-stokes.xml`, for the Navier-Stokes Solver. In addition, we will add the Extra EDF, `saveline.xml`, for the SaveLine Solver.

Finish by clicking on OK. ElmerGUI will generate a mesh which is visualized in Figure 20.3.

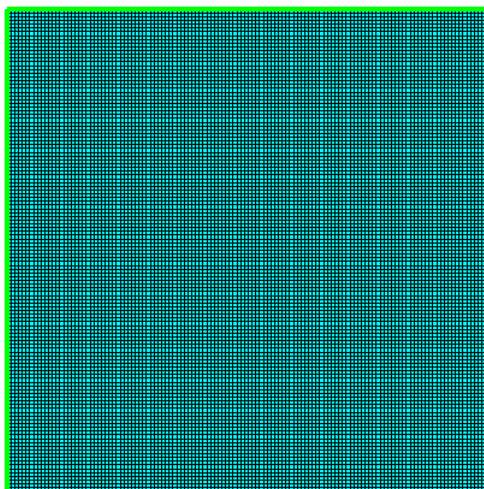


Figure 20.3: The mesh for the driven cavity problem

You can verify that the mesh consists of 16384 surface elements by inspecting the dialog box that appears when selecting

```
Model
  Summary
```

We can now define the model going through the Model menu from the top to bottom. In the equation section we choose the relevant equations and parameters related to their solution. We have only one set of equations which consists of the incompressible Navier-Stokes equation. We also will use the SaveLine tab in the equations menu.

```
Model
  Equation -> Add ..
```

We select the Navier-Stokes tab, click on the ‘Active’ box, and click to apply the equation to ‘Body 1’. We will accept the default settings for the Navier-Stokes equation for now.

Since we have two solvers active for Body 1, you may ask, which solver will run first and which solver will run second? Elmer has the ability to control when each solver will run. For example, we could add a priority number to each solver, where the higher priority number solver will run first. In this case, we want the Saveline solver to run after the Navier-Stokes solver has run. While on the Navier-Stokes tab, enter ‘2’ in the priority box, then switch to the Saveline tab and enter ‘1’ in the priority box. Now the Navier-Stokes solver has the higher (numerically larger) priority number, so it will execute before the Saveline solver.

While on the SaveLine tab, click on the ‘Active’ box. Verify that ‘Body 1’ is already selected. Don’t click on the ‘OK’ button just yet, we still need to add some info into the Saveline solver.

The equation tabs should look like Figure 20.4.

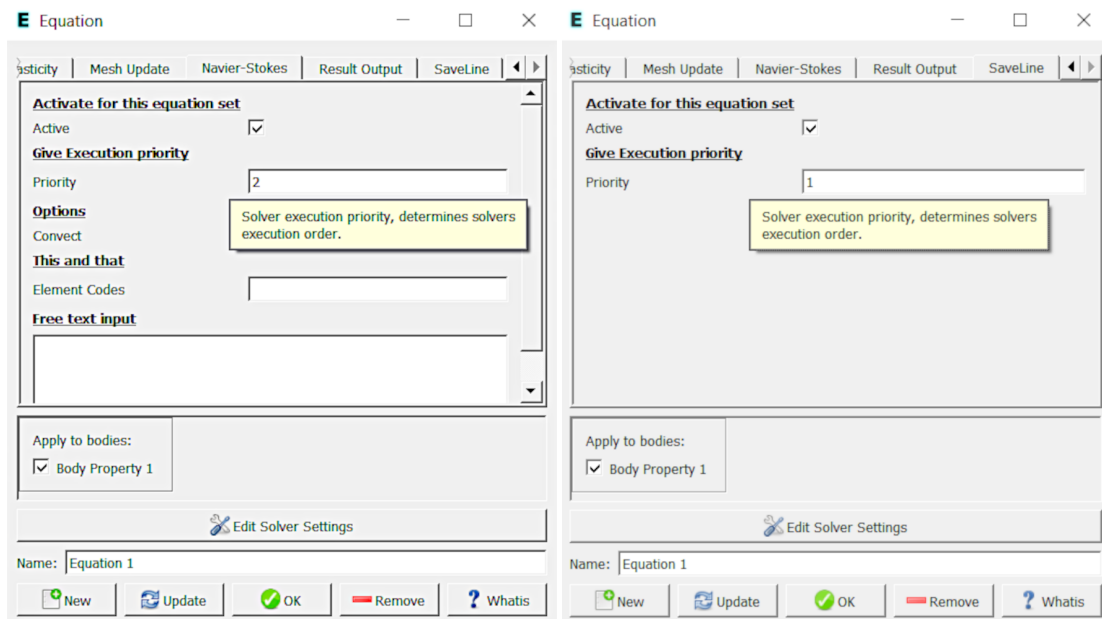


Figure 20.4: Activate Equations, Navier-Stokes on left, Saveline on right

While the SaveLine tab is active, click on the ‘Edit Solver Settings’ tab, followed by the ‘Solver specific options’ tab, and enter the information below as shown. Saveline needs to know what name to use when it creates the results file, in this case, we will enter ‘save-400.dat’. For Polyline Coordinates and for Polyline Divisions, enter

```
Size(4,2); 0.5 1.0 0.5 0.0 1.0 0.5 0.0 0.5
Size(2);20 20
```

The SaveLine tab should look like Figure 20.5. Now finish up by clicking on the ‘OK’ button to close the Equation menu.

In a later section we will discuss the details of creating the SaveLine settings.

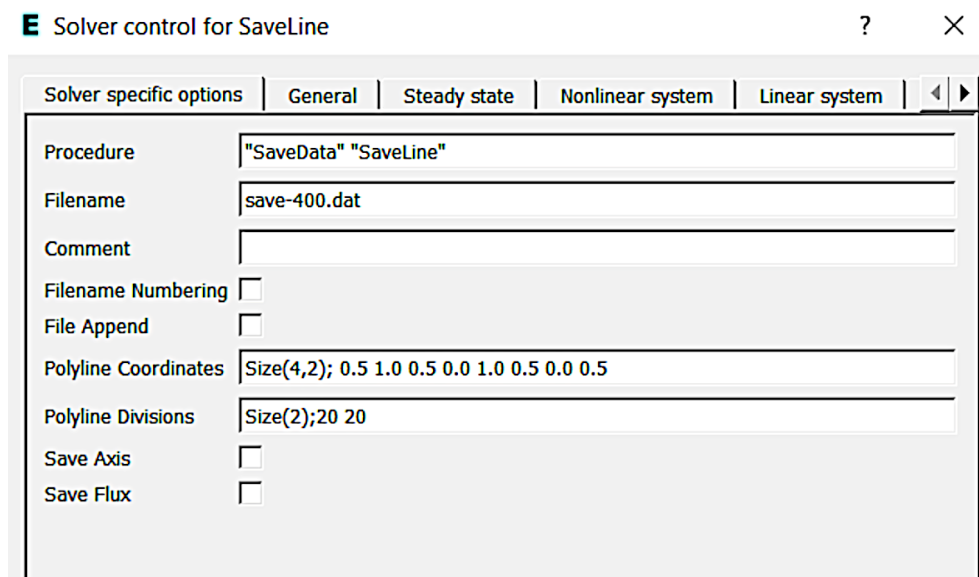


Figure 20.5: SaveLine Specific Settings

The Material section includes all material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the density. Other properties assume a physical law, such as the viscosity. In the material section, we select

Model

Material -> Add ..

The Material menu allows entry of any desired material properties, or one could select pre-defined materials from the 'Material Library' button. In our case, we will enter the material values as stated under 'Case Definition' above.

In the dialog box that appears, we first click on the General tab and we set the density equal to 1 and apply it to "Body 1" as shown in left panel of Figure 20.6. Subsequently, we select the "Navier-Stokes" tab where we set the viscosity to 0.01, the "Compressibility model" to "Incompressible", apply it to "Body 1" and click ok as shown in the right panel of Figure 20.6.

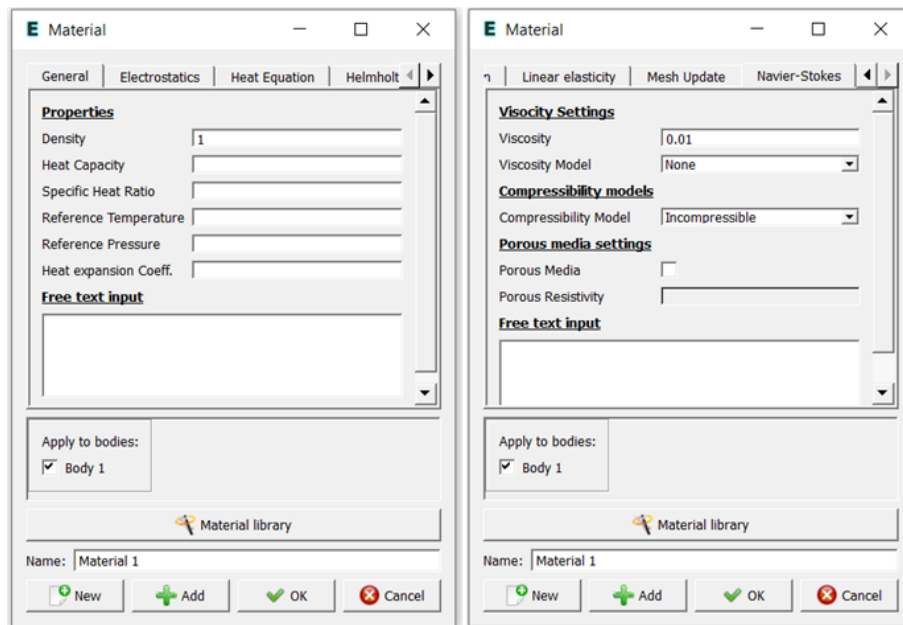


Figure 20.6: Material properties

For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1,...) or direct Umfpack solver, for example.

The appropriate boundary conditions can be set via

Model

Boundary condition -> Add ..

In the dialog box that appears, we select the "Navier-Stokes" tab. For boundary condition 1, we apply the no slip condition on boundaries 1, 2 and 4 by clicking on the "No wall slip BC" option in the Navier-Stokes tab as shown in the left panel of Figure 20.7. For boundary condition 2, we apply the lid velocity with 4 m/s in the x-direction and 0 m/s in the y-direction in the same tab after selecting "Boundary 3" as shown in the right panel of Figure 20.7.

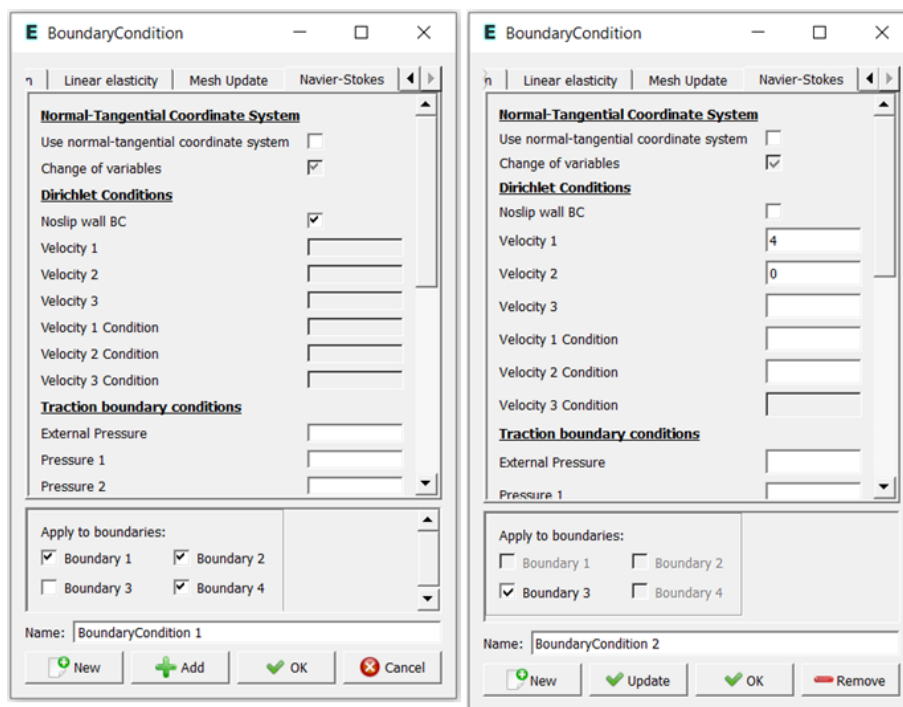


Figure 20.7: Boundary conditions for the driven cavity

For the execution, ElmerSolver needs the mesh files and the command file. We can write and view the command file by selecting

```
Sif
Generate
Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The ElmerGUI project includes all the files needed to restart the case.

```
File
Save Project
```

After we have successfully saved the files we may start the solver.

```
Run
Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. The problem should converge in less than ten iterations. The variation of the residuals should be similar to the one in Figure 20.8. The required computational time should be a couple of minutes.

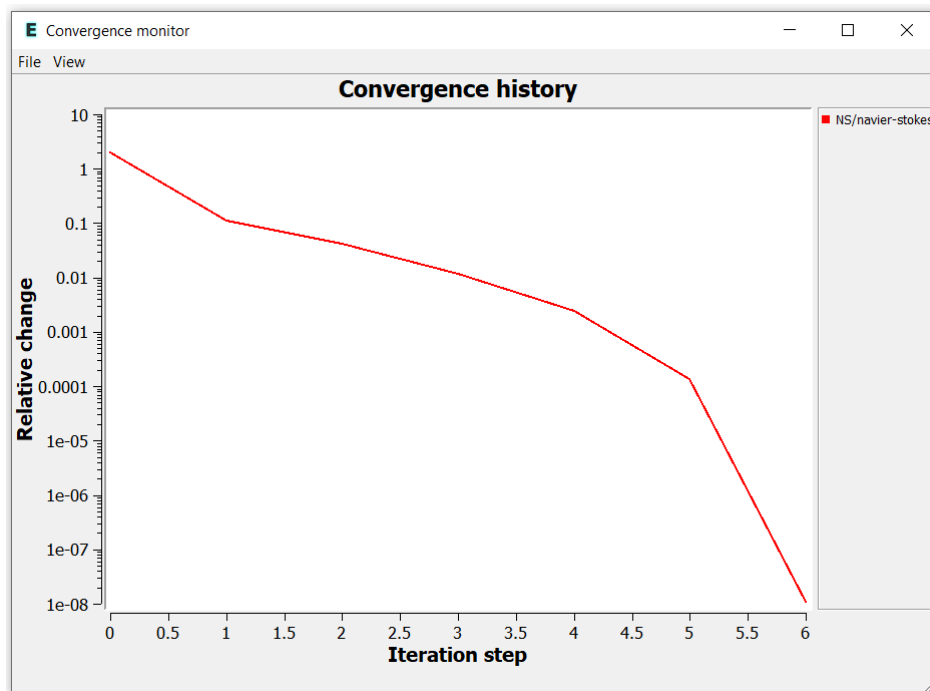


Figure 20.8: Convergence of the solver.

When there are some results to view we may start the postprocessor:

```
Run
Start ParaView
```

or

```
Run
Start ElmerVTK
```

Graphical Results

You may visualize the results with Paraview and/or ElmerVTK. Results from each of the postprocessors is shown, with the Paraview results on the left and the ElmerVTK results on the right. In Figure 20.9 the magnitude of the obtained velocity field is presented. The x- and y- cartesian components of the velocity vectors are shown in Figure 20.10 and 20.11, respectively.

The results for the case of Reynolds number = 400 are shown in Figure 20.9, Figure 20.10, and Figure 20.11.

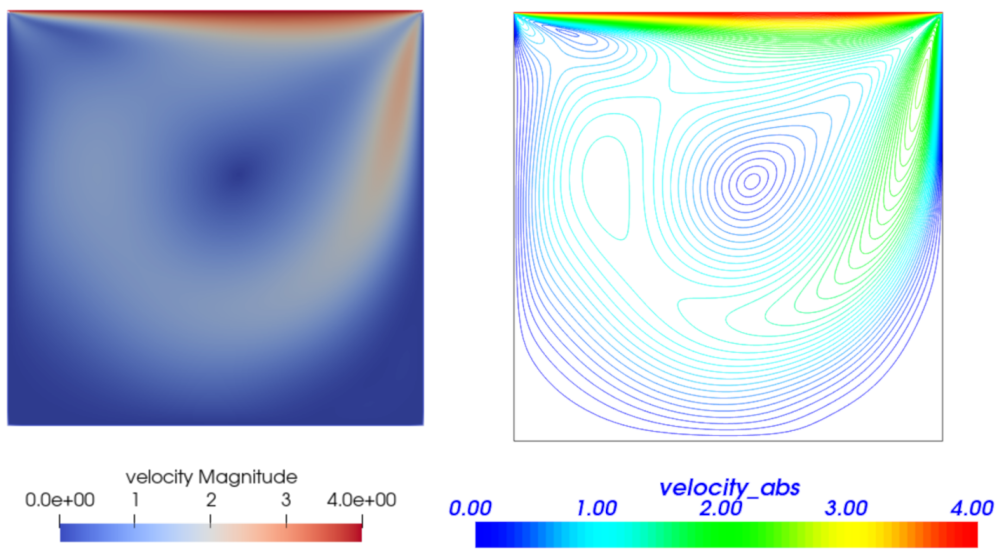


Figure 20.9: $Re = 400$: Distribution of the magnitude of the velocity vector.

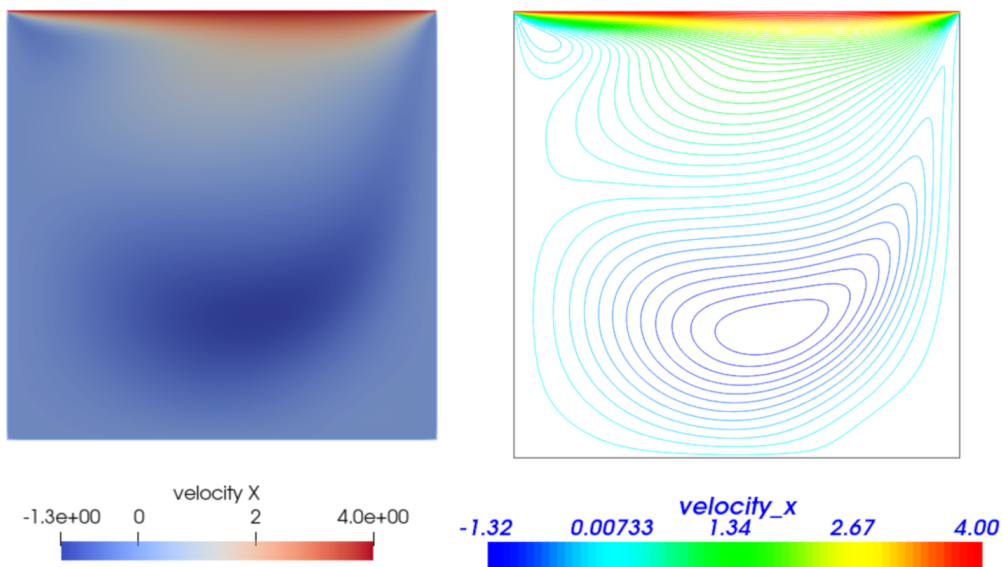


Figure 20.10: $Re = 400$: Distribution of x component of the velocity vector.

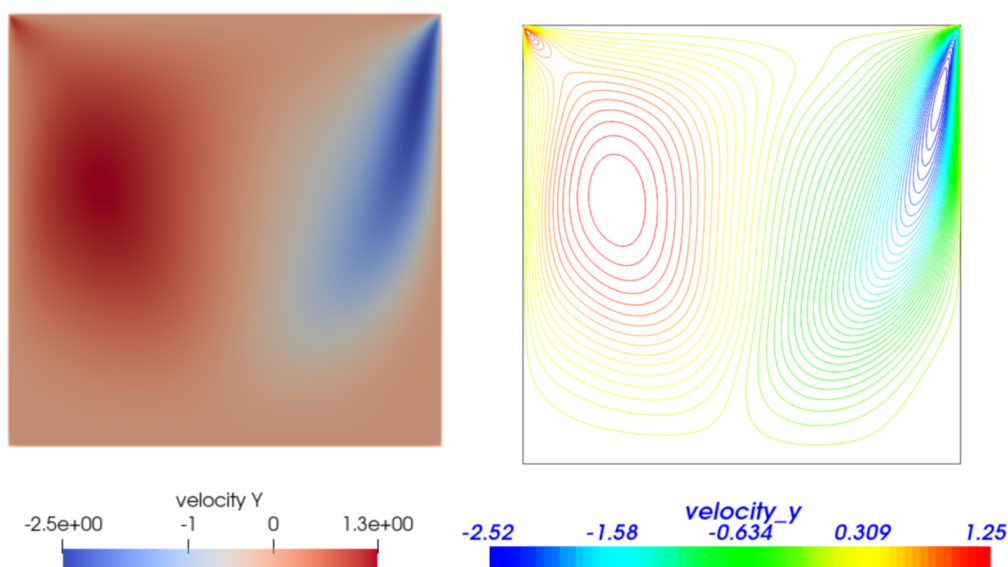


Figure 20.11: Re 400: Distribution of y component of the velocity vector.

SaveLine

The SaveLine subroutine contained in the SaveData solver is used to save Elmer solution data in ASCII format along a line segment running through a 2D or 3D geometry model. The SaveLine subroutine and keywords are described in detail in the Elmer Models Manual.

The ElmerGUI equation tab for SaveLine was previously shown in Figure 20.5. We entered data in several of the fields, such as Filename, Polyline Coordinates, and Polyline Divisions.

For Filename, enter a descriptive name such as 'save-400.dat', which indicates the Reynolds number 400 case, with an extension such as 'dat' or 'csv'. The file name and extension can be freely chosen.

Filename Numbering and File Append are not used for this case, but can be used when multiple simulation runs are performed, such as a transient simulation.

Polyline Coordinates and Polyline Divisions are described in the Elmer Models Manual as follows:

Polyline Coordinates(n,DIM) Real

Save the line consisting of line segments defined by two points ($n = 2$).

There can be more than one set of points ($n = 2; 4; 6; \dots$) but as a line segment is defined by two points there must be an even number of points.

Polyline Divisions(n/2,DIM) Integer

The user may give the number of divisions for each polyline. This allows also the proper saving of discontinuous data. The size of this vector should be such that it is compatible with the number of lines.

Polylines could also be called 'Multiple Line Segments', because two end points must be defined for each line segment. For one line segment, two points, for two line segments, four points, etc. Each point has the number of coordinates defined by the simulation, two coordinates for 2D and three coordinates for 3D.

In this case, we are dealing with a 2D simulation, so there will be two coordinates with $DIM = 2$. The Ghia, et al, article presents tabular data for a vertical line and a horizontal line, both of which pass through the geometric center. We will need to define two line segments with four points, so $n = 4$. If we multiply $n * DIM$, ($4*2=8$) then we get the number of individual real numbers that must be listed.

Refer back to Figure 20.5 and in the Polyline Coordinate entry box you will see 'Size(4,2);' which defines 'n' and 'DIM', followed by a series of eight numbers.

The Polyline Divisions entry box contains 'Size(2);', where we enter 'n/2', ($4/2=2$), followed by two numbers, one for each line segment. Each number indicates how many division are desired for each line segment. So if we want data results printed for a desired number of points along each line segment, enter that number here. In this case, we want twenty points along each line segment. On the other hand, if there is a sharp curve in the line segment that we want to capture, dividing the line segment into 50 or 100 pieces, or splitting the single line segment into three line segments with different divisions, will help add detail when we want to graph the results.

SaveLine Results

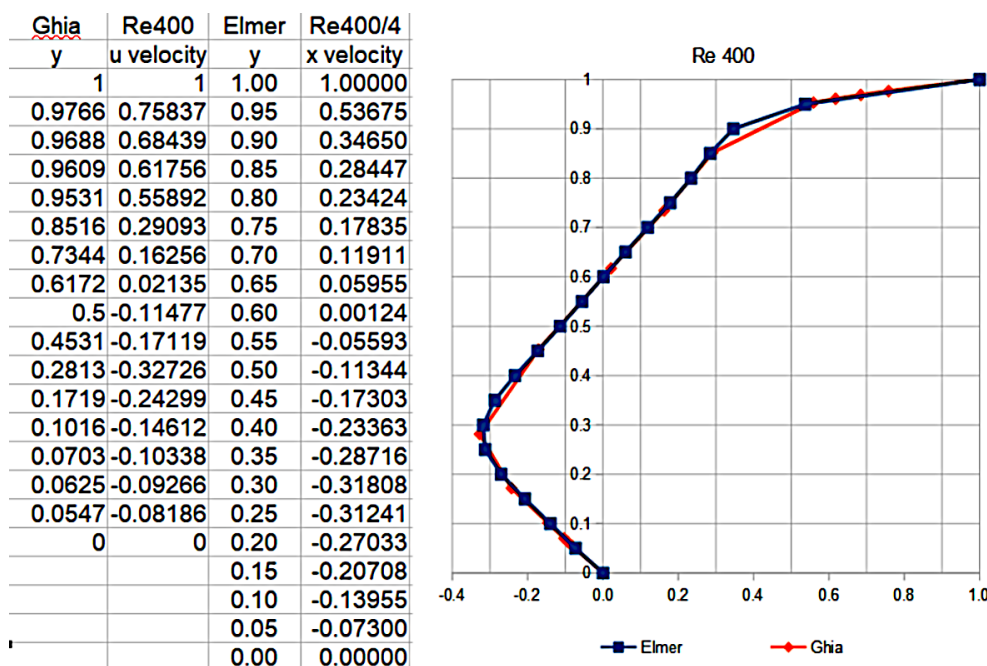


Figure 20.12: Results comparison for $Re = 400$

The left side of the spreadsheet presents the u velocity data from Table 1 in Ghia, et al, and the right side of the spreadsheet presents the results calculated by Elmer.

The Elmer results must be normalized to between 0.0 and 1.0, by dividing by the driven wall velocity, in this case divided by 4.

There is decent agreement between the two sets of results with a Reynolds number of 400.

High Reynolds Numbers

The article by Ghia, et al, presents data over a range of Reynolds number. The actual values in the article are 100, 400, 1000, 3200, 5000, 7500, and 10,000. You may try to solve the cases with parameters corresponding to the given list of Reynolds numbers. This can be achieved by modifying the velocity of the upper boundary from 1 m/s up to 100 m/s, respectively.

Furthermore, although not shown in this tutorial, you can compare the obtained results with the data from the NASA repository at <https://www.grc.nasa.gov/www/wind/valid/cavity/cavity.html>.

When running the range of Reynolds numbers, one will notice that starting at 50 m/s driven velocity the solution will no longer converge when using the default solver settings. Convergence can be achieved up to the maximum speed of 100 m/s, Reynolds number of 10,000, by changing the Navier-Stokes solver settings.

The Elmer Models Manual describes the details behind each of the documented solvers. The Navier-Stokes Flow Solver is used for this driven cavity tutorial. Increasing the Reynolds number to 10,000 leads to issues with convergence. The Linearization section of the Flow Solver chapter mentions: ‘... to first take a couple of Picard iterations, and switch to Newton iteration after the convergence has begun.’ In this case, increasing the number of Picard iterations from 3 to 20, allows for convergence of the solution. One other setting change is needed for convergence in this situation, and that is to change from iterative to direct (umfpack) on the Linear tab.

Model

Equation

Equation 1

select the Navier-Stokes tab

Edit Solver Settings

select the Linear system tab

click on the direct button under Method
 select Umfpack as the direct solver
 select the Nonlinear system tab
 Newton After iterations
 change from 3 to 20
 Apply
 Update
 OK

For the case of Reynolds number = 10,000, velocity profiles are shown in Figures 20.13, 20.14 and 20.15.

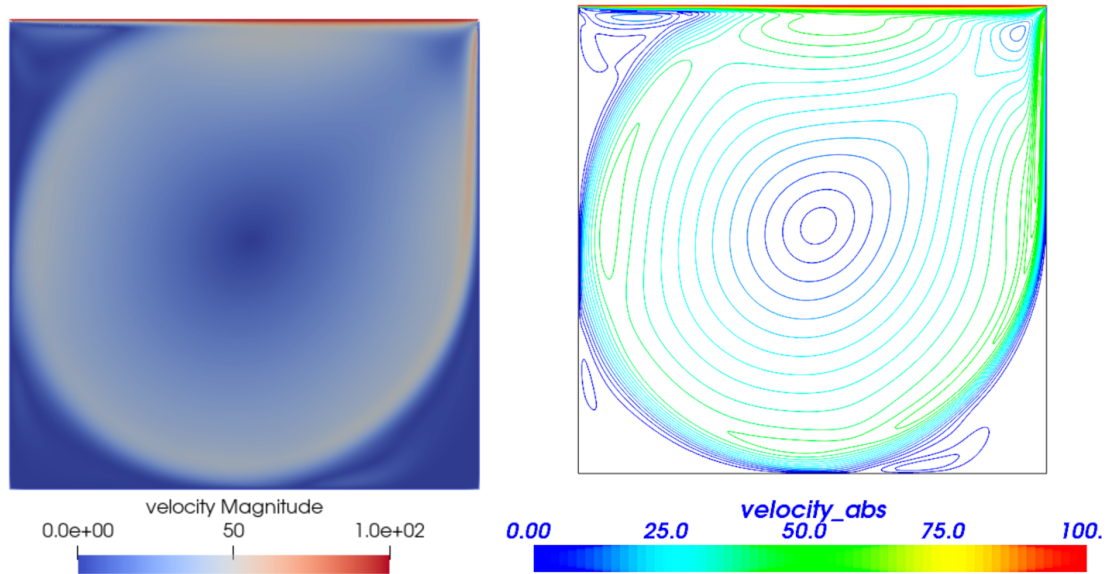


Figure 20.13: Re 10,000: Distribution of the magnitude of the velocity vector.

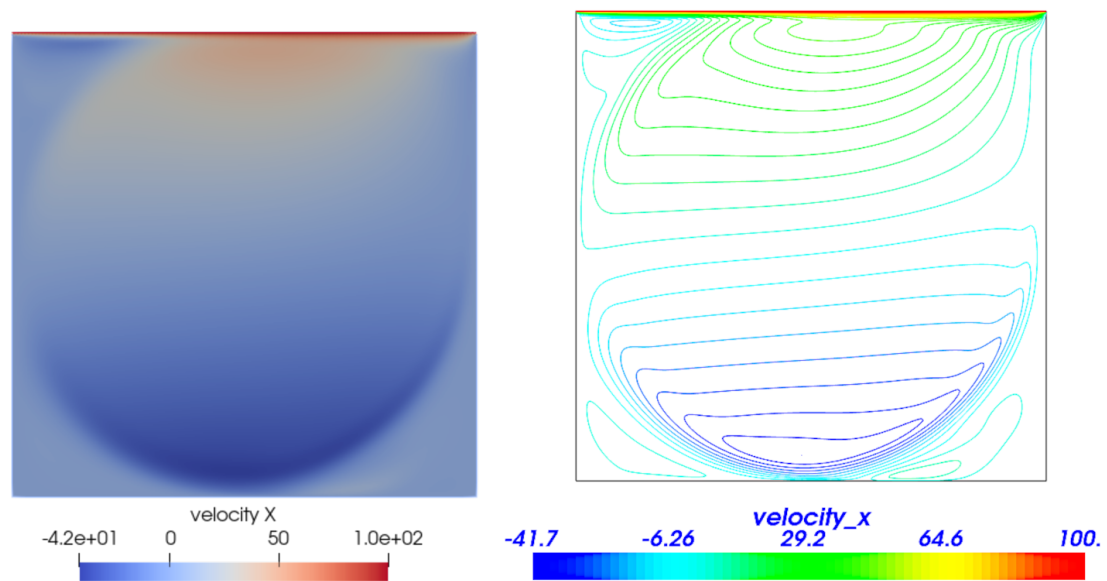


Figure 20.14: Re 10,000: Distribution of x component of the velocity vector.

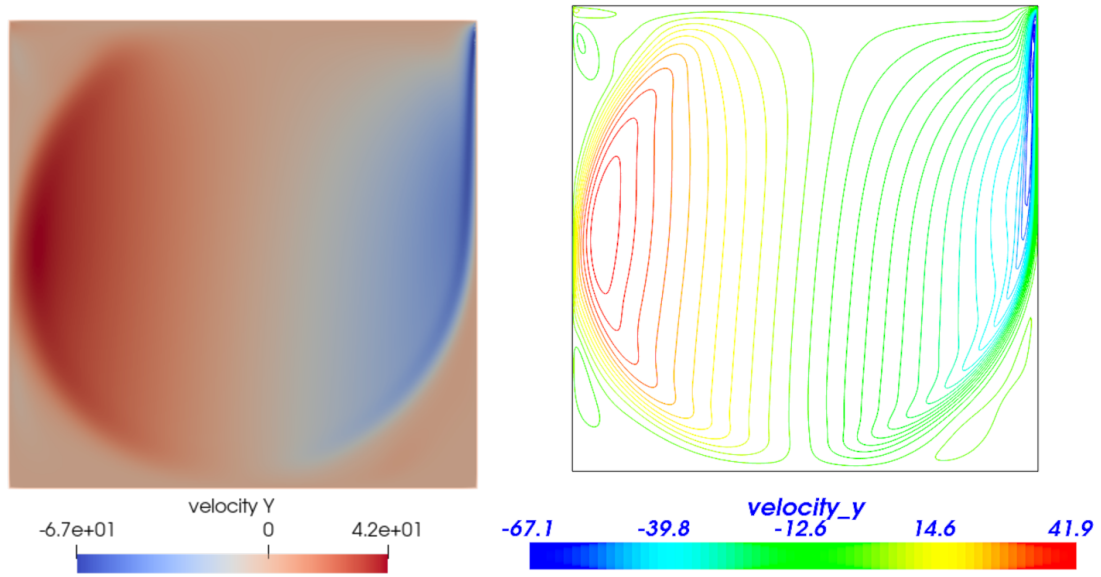


Figure 20.15: Re 10,000: Distribution of y component of the velocity vector.

Saveline results for the case with Reynolds number = 10,000 are shown in Figures 20.16 and 20.17. Some deviation between Elmer results and Ghia results can be observed at the high Reynolds number case. Overall, the shape of the curves of the results show good agreement.

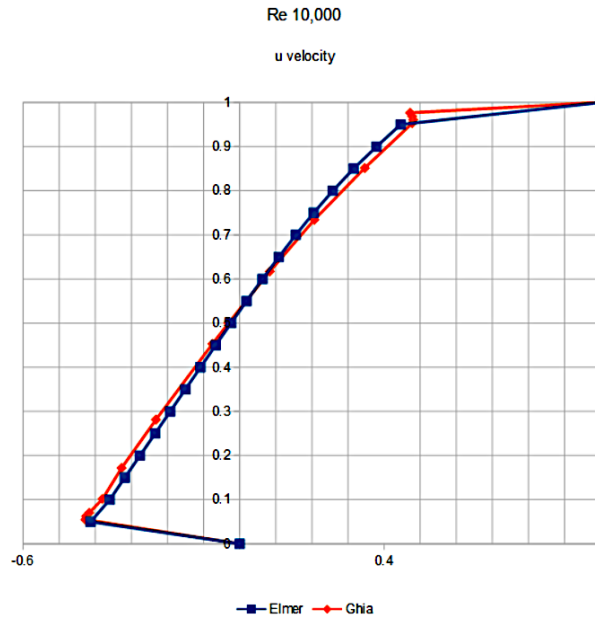


Figure 20.16: Re 10,000 u velocity profiles

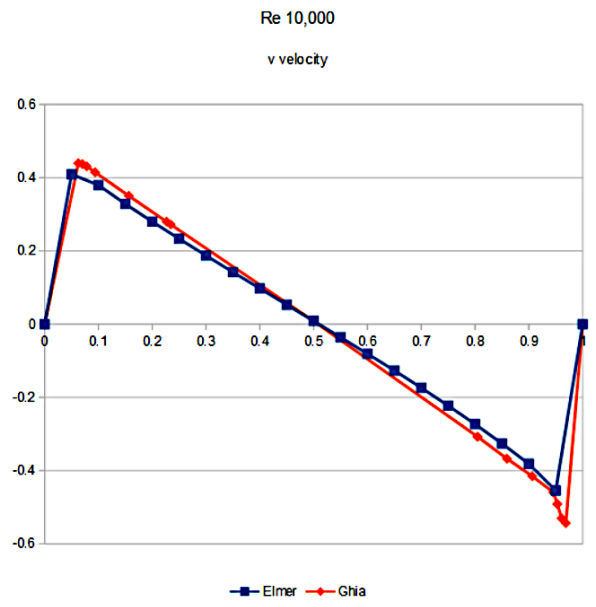


Figure 20.17: Re 10,000 v velocity profiles

Extra task:

For additional credit, try increasing the Reynolds number above 10,000. For example, the solution will converge at Re = 20,000, if the non-linear relaxation factor is reduced from 1.0 to 0.7.

Tutorial 21

Navier-Stokes equation – Turbulent incompressible flow passing a step

Directory: FlowStepKe

Solvers: FlowSolve, KESolver

Tools: ElmerGUI

Dimensions: 2D, Steady-state

Author: Peter Råback, Juha Ruokolainen

Case definition

This tutorial is a natural continuation of the tutorial 19 where the same case was solved with a smaller Reynolds number. It is advisable to study that case before working with this case.

When the Reynolds number increases, the Navier-Stokes equations do not possess any steady-state solution. Basically the solution can be averaged simulating the transient flow over time. However, the computational cost of this approach is often very heavy particularly while at high Reynolds numbers the computational mesh in direct numerical simulation needs to be very dense. Instead its customary to solve time averaged equations. Unfortunately these equations include unknown correlations between quantities that need to be modelled in some way.

The workhorse of turbulence modelling is the $k - \varepsilon$ model which is used in this tutorial. The $k - \varepsilon$ model is a two-equation model that introduces two additional variables – the turbulent kinetic energy k and the turbulent dissipation ε which determines the scale of the turbulence.

The case under study is the canonical step flow of viscous fluid. A fluid, flowing past a step has the density 1 kg/m^3 and viscosity $1.0e - 4 \text{ kg/ms}$. The velocity profile at the inlet is defined by a parabolic profile with mean velocity $v_x = 1.0 \text{ m/s}$ and $v_y = 0.0 \text{ m/s}$. This way the Reynolds number will be 10000. At the outlet only the vertical component is defined, $v_y = 0.0 \text{ m/s}$. At all other walls the no-slip boundary condition, $\vec{v} = 0$, is applied.

Also the new turbulent variables require boundary conditions. In the inlet the condition should reflect the values for developed turbulent profile. Here we roughly estimate the turbulent kinetic energy and elongate the inlet distance before the step to have a fully developed turbulent profile. From the literature it is the turbulent intensity i.e. the kinetic energy of turbulence vs. the kinetic energy of mean flow scales as $0.16\text{Re}^{-1/8}$. For our current configuration an estimate for the turbulent kinetic energy is 0.00457. For the walls a no-slip condition is applied which also sets the values of the turbulent parameters accordingly. Also a boundary layer model could be used but here our mesh should be able to capture even the boundary phenomena quite well.

Solution procedure

The definitions for the relevant equation are not loaded into ElmerGUI by default. Hence, one needs to load these before starting the simulations.

File

Definitions

Append -> k-epsilon.xml

The additional definitions should reside in the directory `edf-extra` within the distribution.

Optionally, one may copy the desired `xml` files to the `edf`-directory from the directory `edf-extra` which will enable automatic loading of the definitions every time ElmerGUI is started. By inspecting the definitions in the Elmer Definitions File editor one may verify that the new definitions were really appended.

The mesh is given in ElmerGrid format in file `steplong.grd`, load this file.

File

```
Open -> steplong.grd
```

You should obtain your mesh and may check that it consists of 14584 nodes and of 14175 bilinear elements.

Model

```
Summary...
```

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation. The steady-state simulation is carried out in 2-dimensional Cartesian coordinates, which are also the defaults. The coupled system converges unfortunately quite slowly and hence we need to increase the number of maximum iterations.

Model

```
Setup
  Simulation Type = Steady state
  Coordinate system = Cartesian
  Steady state max. iter = 100
```

In the equation section we choose the relevant equations and parameters related to their solution. In this case the Navier-Stokes and $k - \varepsilon$ equations are needed. We want to solve the Navier-Stokes and $k - \varepsilon$ equations iteratively using only one non-linear iteration for optimal convergence of the coupled system. Some relaxation is needed in order to achieve convergence at all. We also relax a little bit on the steady state convergence tolerance. Initially the Navier-Stokes solver uses the more robust Picard iteration which may be changed to Newton iteration after the iteration progresses. However, here we want to suppress the use of Newton linearisation since it seems to cause problems with the $k - \varepsilon$ equation.

Model

```
Equation
  Name = Flow equations
  Apply to Bodies = Body 1
  Navier-Stokes
    Active = on
    Edit Solver Setting
      Nonlinear System
        Max. iterations = 1
        Relaxation factor = 0.5
        Newton after tolerance = 0.0
      Steady state
        Convergence tol. = 1.0e-4
  K-Epsilon
    Active = on
    Edit Solver Setting
      Nonlinear System
        Max. iterations = 1
        Relaxation factor = 0.5
      Steady state
        Convergence tol. = 1.0e-4
  Add
  OK
```

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the density. Other properties assume a physical law, such as viscosity. For the model parameters of the turbulent equations we are happy with the defaults.

```

Model
Material
  Name = Ideal
  General
    Density = 1.0
  Navier-Stokes
    Viscosity = 1.0e-4
    Viscosity Model = K-Epsilon
  Apply to Bodies = Body 1
Add
OK

```

The current case does not have any body forces. To help in the convergence we make a rude initial guess.

```

Model
Initial Condition
  Name = Initial Guess
  Navier-Stokes
    Velocity 1 = 0.0
    Velocity 2 = 0.0
  K-Epsilon
    Kinetic Energy = 0.00457
    Kinetic Dissipation = 1.0e-4
  Apply to Bodies = Body 1
Add
OK

```

When defining Boundary conditions it is possible to assign to the boundaries immediately, or to use mouse selection to assign them later. In this case we use the latter approach since we do not necessarily know the numbering of boundaries by heart. There is a special boundary condition that takes care of the Boundary conditions for the no-slip walls for both the Navier-Stokes and $k - \varepsilon$ equation. Additionally there are inlet and outlet conditions. For the inlet click Enter to open an edit box for the Velocity 1 when typing in the expression, which will be evaluated at run-time so that $v_x = 6(y - 1)(2 - y)$.

```

Model
BoundaryCondition
Add
  Name = Inlet
  Navier-Stokes
    Velocity 1 = Variable Coordinate 2; Real MATC "6*(tx-1)*(2-tx)"
    Velocity 2 = 0.0
  K-Epsilon
    Kinetic Energy = 0.00457
    Kinetic Dissipation = 1.0e-4
Add
New

  Name = Outlet
  Navier-Stokes
    Velocity 2 = 0.0
Add
New

  Name = Walls
  Navier-Stokes
    Noslip Wall BC = on
Add

```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

Model

```
Set boundary properties
  Choose inlet -> set boundary condition Inlet
  Choose outlet -> set boundary condition Outlet
  Choose walls -> set boundary condition Walls
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

Sif

```
Generate
Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case. Create a suitable directory for the case if needed.

File

```
Save Project
```

After we have successfully saved the files we may start the solver

Run

```
Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. The convergence is far from monotonic but computation should terminate when sufficient convergence is reached after 30 iterations.

Results

When there are some results to view we may start Paraview for the postprocessing.

Run

```
Paraview
```

The results may be viewed using the postprocessor as shown in Figures 21.1 and 21.2. One may also register specific values, for example the pressure difference is 0.302 Pa, the minimum horizontal and vertical velocities are -0.213 m/s and -0.0834 m/s, respectively. One special result of interest is the point, on the x-axis, at which the direction of the flow changes. In this case its position is about 5.1 m after the step.

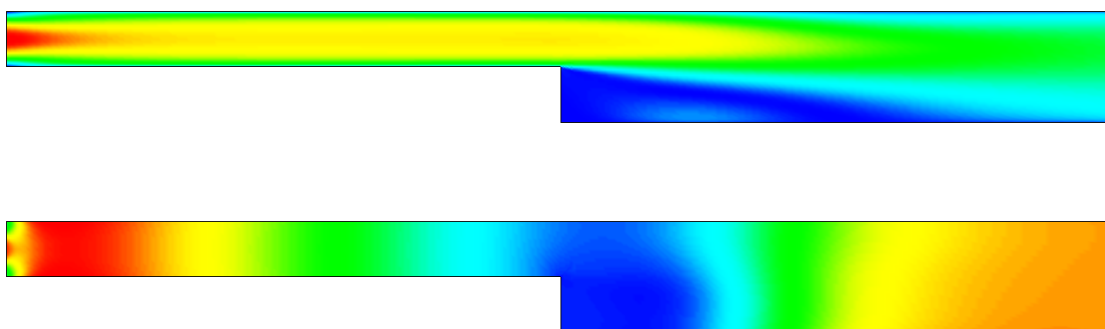


Figure 21.1: Variables of the Navier-Stokes solver: absolute velocity on top and pressure on bottom

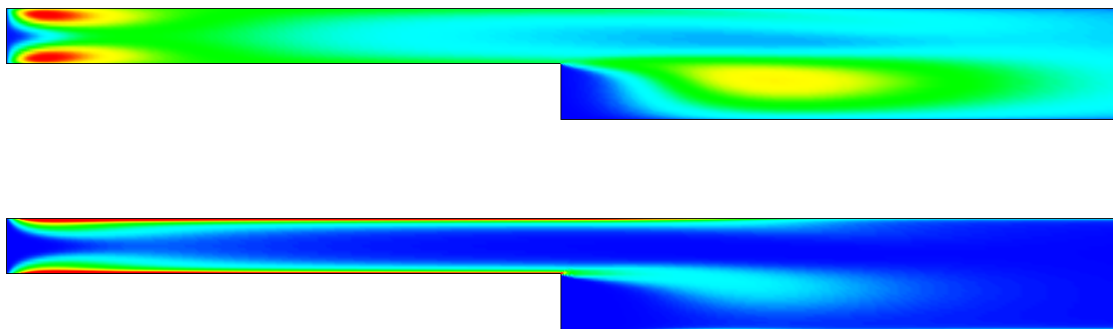


Figure 21.2: Variables of the $k - \epsilon$ solver: kinetic energy on top and its dissipation on bottom

Tutorial 22

Navier-Stokes equation – Laminar compressible flow over a step

Directory: FlowStepCompressible

Solvers: HeatSolve, FlowSolve

Tools: ElmerGUI

Dimensions: 2D, Steady-state

Author: Juha Ruokolainen, Rich Bayless

Case definition

This tutorial demonstrates how to simulate compressible air flowing past a step. The whole step has length of 1.4 m and the height of 0.2 m and the first part of it has length of 0.4 m and the height of 0.1 m (Figure 22.1). The model has three sets of boundary conditions. The air flows into the step from the inlet region and withdraws from the outlet region. The other edges of the step compose the third boundary.

The flowing air is considered as an ideal gas in this case, and its density ρ depends on the pressure p and temperature T through the equation of state where R is the gas constant. The following table lists the material parameters for the gas used in this tutorial. Note that the parameters being used in this tutorial are different from the parameters that would be retrieved from the Material Library, if one selected the entry for 'Air (Room Temperature).

$$\rho = \frac{p}{RT},$$

Table 22.1: Material parameters.

parameter	value
viscosity	16.7e-6 Ns/m ²
heat conductivity	0.026 W/(m·K)
heat capacity	1.01e3 J/(kg·K)
specific heat ratio	1.4
reference pressure	1e5 Pa

The geometry looks as shown in figure 22.1

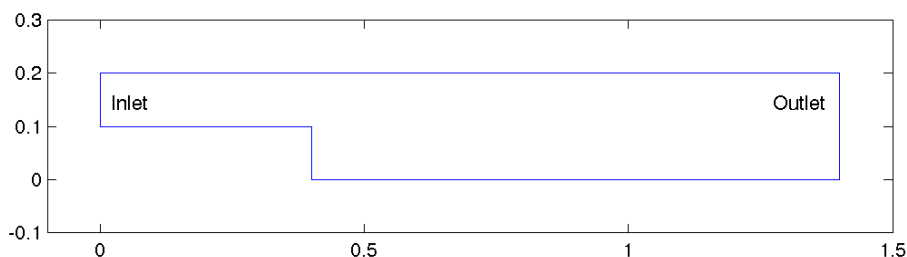


Figure 22.1: Geometry of the Step.

Check ElmerGUI Equation Menu

We will be using the FlowSolver and the HeatSolver, which are the default, pre-loaded GUI definitions, so they don't need to be manually activated. For reference, the `heatequation.xml` and `navier-stokes.xml` definition files are, by default, located in Linux in:

```
$ELMER_HOME/share/ElmerGUI/edf
```

and in Windows, located in:

```
C:/Program Files/Elmer 9.0-Release/share/ElmerGUI/edf
```

Solution procedure

The mesh is given in ElmerGrid format in file `mesh.grd`, load this file.

File

```
Open -> mesh.grd
```

You should obtain your mesh and may check `Model Summary...` that it consists of 531 surface elements. Your mesh should look like as shown in figure 22.2.

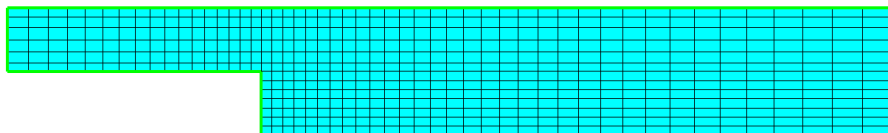


Figure 22.2: Mesh

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation uses 2D Cartesian geometry and the problem is solved in steady state using no more than twenty steady state iterations.

Model

Setup

```
Simulation Type = Steady State
```

```
Steady state max. iter = 20
```

Apply

In the equation section we choose the relevant equations and parameters related to their solution.

In this case we'll have one set of equations (named "Heat and Flow") which consists of the Heat equation and of the Navier-Stokes equation.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and therefore its easier to assign the Equation and Material to it directly. It is important to select the convection to be computed since that couples the velocity field to the heat equation.

For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1,...) or direct Umfpack solver, for example.

```

Model
Equation
Name = Heat and Flow
Apply to Bodies = 1
Heat Equation
Active = on
Convection = Computed
Edit Solver Settings
General
Numerical techniques
Stabilize = uncheck
Bubbles = check
Apply (to exit Edit Solver Settings)
Navier-Stokes
Active = on
Edit Solver Settings
General
Numerical techniques
Stabilize = uncheck
Bubbles = check
Apply (to exit Edit Solver Settings)
Add
OK

```

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such as conductivities and viscosity.

Our intention is to model compressible flow and that is why we have to set the value "Perfect Gas" for the Compressibility Model entry. Furthermore, because perfect gas model has been chosen the settings Reference Pressure and Specific Heat Ratio must also be given. The Navier-Stokes equation also needs the value of viscosity and the heat equation needs the values of heat capacity and heat conductivity. We will not use the Material Library, instead we will enter our own values. The needed material parameters for an ideal gas for this tutorial are listed in table 22.1.

```

Model
Material
Name = Our Ideal Gas
Apply to Bodies = 1
General
Heat Capacity = 1.01e3
Specific Heat Ratio = 1.4
Reference Pressure = 1.0e5
Heat Equation
Heat Conductivity = 0.026
Navier-Stokes
Viscosity = 16.7e-6
Compressibility models = Perfect Gas
Add
OK

```

A Body Force represents the right-hand-side of a equation. It is generally not a required field for a body.

Initial conditions should be given to transient cases, and probably are not needed for steady state solutions. For the initial condition of temperature we have chosen 300 K.

```

Model
Initial Condition
Name = Initial Guess
Apply to Bodies = 1
Heat Equation

```

```

Temperature = 300
Add
OK

```

Only one boundary condition may be applied to each boundary and therefore all the different physical BCs for a boundary should be grouped together.

There are three sets of boundary conditions, so three `Boundary Condition` sections are needed. The first one is used to prescribe the boundary conditions in the inlet region. Note that we have defined the x-velocity and temperature as a variable of y-coordinate. This is done by setting different values for the x-velocity and temperature (the numerical values of the second column between the words `Real` and `End`) in the different y-points (the numerical values of the first column between words `Real` and `End`) of the inlet region. This kind of procedure prevents singularities from occurring in the corner points of the inlet region. In addition, this kind of definition is more realistic than a constant condition, in which the values of the x-velocity and temperature remain the same in the whole inlet region.

To enter the three pairs of numbers in the Heat Equation menu tab, click into the box labelled `Temperature` under `Dirichlet Conditions`, then press the enter key. An editor window will pop up and you can enter all of the text from ‘Temperature’ to ‘Navier-Stokes’ into the window. Repeat the same steps in the Navier-Stokes menu tab, and enter all the text from ‘Velocity 1’ to ‘Velocity 2’. Refer to Figure 22.3, where the left side shows the Temperature boundary condition for Heat and the right side shows the Velocity 1 boundary condition for Flow.

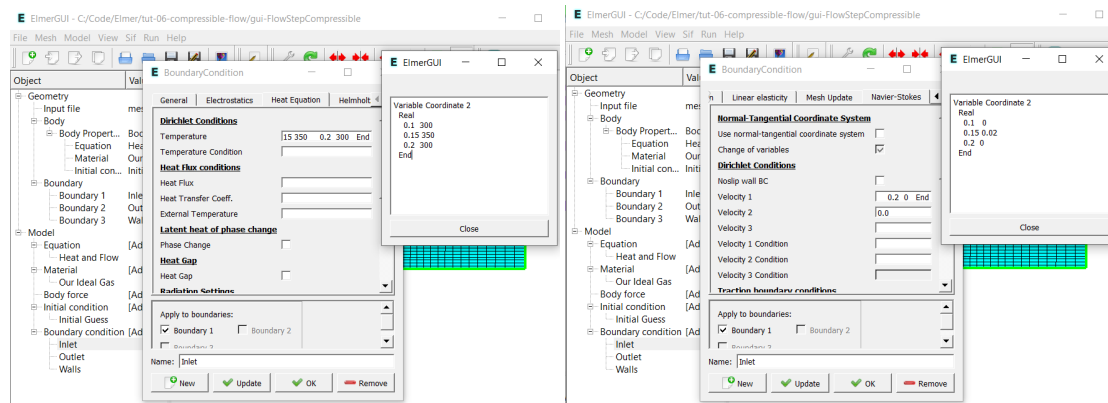


Figure 22.3: Left side: Heat BC, Right side: Flow BC

```

Model
  BoundaryCondition
    Name = Inlet
    Heat Equation
      Temperature = Variable Coordinate 2
      Real
        0.1    300
        0.15   350
        0.2    300
      End
    Navier-Stokes
      Velocity 1 = Variable Coordinate 2
      Real
        0.1    0
        0.15   0.02
        0.2    0
      End
      Velocity 2 = 0.0
    Add
  New

```

After the rest of the boundary conditions have been defined the problem is ready to solve.

```

Name = Outlet
Navier-Stokes
  Velocity 2 = 0.0
Add
New

Name = Walls
Heat Equation
  Temperature = 300
Navier-Stokes
  Noslip wall BC = on
Add
OK

```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking on each boundary with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

```

Model
  Set boundary properties
    Choose Inlet -> set boundary condition
    Choose Outlet -> set boundary condition
    Choose Walls -> set boundary condition
  OK

```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```

Sif
  Generate
  Edit -> look how your command file came out

```

Before we can execute the solver we should save the files in a directory. The ElmerGUI project includes all the files needed to restart the case.

```

File
  Save Project

```

After we have successfully saved the files we may start the solver

```
Run
Start solver
```

A convergence view automatically pops up showing relative changes of each iteration.
When there are some results to view we may start the postprocessor also

```
Run
Start ParaView
```

Results

The simulation may take about a minute. You may inspect the results with Paraview.

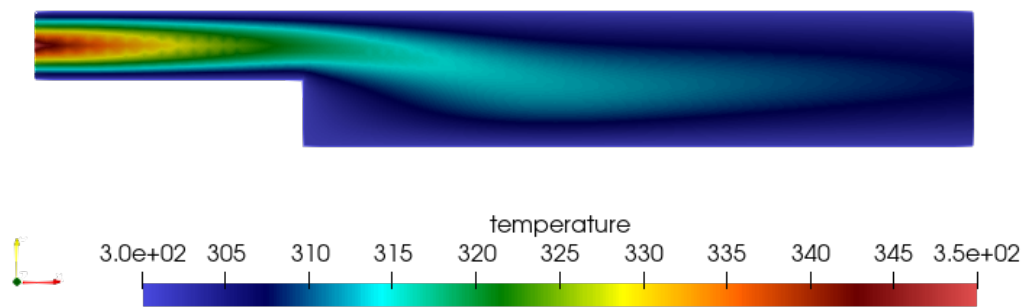


Figure 22.4: Temperature distribution

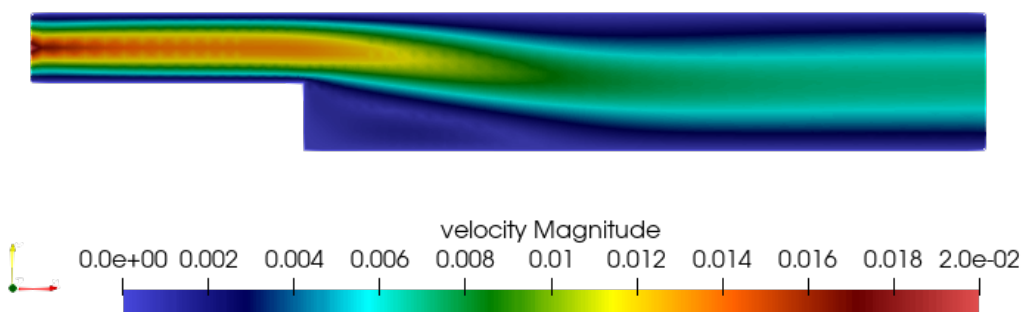


Figure 22.5: Velocity distribution

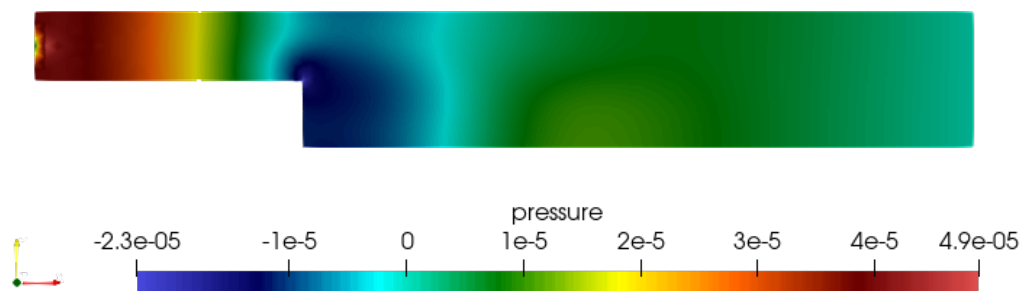


Figure 22.6: Pressure distribution

Extra task:

If you have time you may try to solve the case with different parameters, such as doubling the peak inlet velocity in the inlet boundary condition from 0.02 to 0.04. See how much you can increase the peak inlet velocity and still converge to a solution.

Tutorial 23

Vortex shedding – von Karman instability

Directory: VonKarmanGUI

Solvers: FlowSolve

Tools: ElmerGUI

Dimensions: 2D, Transient

Author: Juha Ruokolainen, Peter Råback

Case definition

This tutorial is about simulating the development of vortex shedding i.e. the von Karman instability. For full details on the problem look at the benchmark case definition in the 1996 article by M. Schäfer and S. Turek, "*Benchmark computations of laminar flow around a cylinder*".

The geometry is a 2D channel with a circular obstacle. The channel height is 0.41 meters and the channel length is 2.2 meters. The circular obstacle is 0.1 meters in diameter, located at $x = 0.2$ meters and $y = 0.2$ meters. The obstacle is slightly off centerline of the channel.

The flow enters from $x = 0$ and exits at $x = 2.2$ meters. The upper and lower walls and the circular obstacle are set to the no-slip condition, $U = V = 0.0$. The inlet flow has a parabolic profile, with a maximum flow velocity at centerline of 1.5 m/s. The mean inlet flow velocity is then 1.0 m/s.

The Reynolds Number equals the mean velocity times the obstacle diameter, divided by the kinematic viscosity, $Re = U * D/\nu$. $Re = 1.0 \text{ m/s} * 0.1 \text{ m}/0.001 \text{ m}^2/\text{s} = 100$. The fluid density equals 1.0 kg/m^3 .

Solution procedure

The mesh is given in 2d netgen format in file `circle_in_channel.in2d`, load this file.

File

```
Open -> circle_in_channel.in2d
```

You should get a mesh consisting of 749 nodes and 1328 triangles, as shown in Figure 23.1. Note the increased mesh density near the circular obstacle.

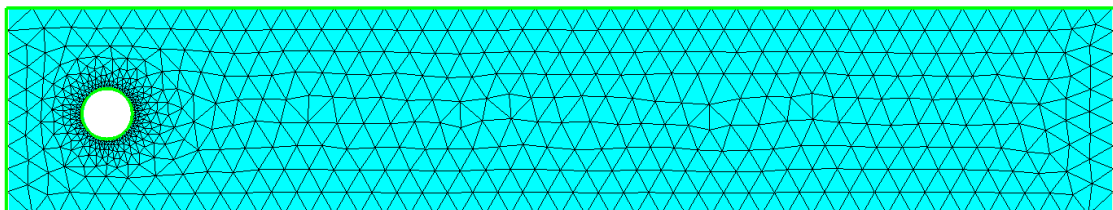


Figure 23.1: Computational mesh of the problem.

This is a rather sparse mesh. To decrease the element size we can refine the mesh using ElmerGUI commands.

```
Mesh
  Configure
    nglib / Max H: 0.02
Mesh
  Remesh
```

This refined mesh now includes 3464 nodes and 6506 triangles.

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried out in 2-dimensional Cartesian coordinates. 2nd order bdf time stepping method is selected with 200 steps and we want the total simulation time to be 8 seconds. Rather than calculating the time step manually, we will use MATC to perform the calculation. MATC statements that start with a '\$' will be evaluated once at the program start. When added to an ElmerGUI project the statement should be entered without internal spaces.

```
Model
  Setup
    Simulation Type = Transient
    Steady state max. iter = 1
    Time Stepping Method = bdf
    BDF Order = 2
    Time Step Intervals = 200
    Time Step Sizes = $8/200
```

For the solver specific settings we are quite happy to use the defaults. However, we relax a little bit the convergence tolerances to get speedier simulation.

```
Model
  Equation
    Name = Navier-Stokes
    Apply to Bodies = 1
    Navier-Stokes
      Active = on
    Edit Solver Settings
      Nonlinear system
        Convergence tol. = 1.0e-4
      Linear System
        Convergence tol. = 1.0e-6
    Add
    OK
```

The Material section includes all the material parameters. Here we choose simple parameters for the academic test case

```
Model
  Material
    Name = Ideal
    General
      Density = 1
    Navier Stokes
      Viscosity = 0.001
    Apply to Bodies = 1
    Add
    OK
```

The system does not need any body forces. We are also happy with the default initial condition of zero and therefore no initial conditions are applied either. Any other initial condition would require the values to be explicitly set.

We have three different kinds of boundaries: inlet, no-slip walls, and outlet. The inlet has a parabolic fully developed laminar profile with a maximum velocity of 1.5 m/s. Additionally for the inlet the vertical velocity component is assumed zero. The circle and the lower and upper walls are given the no-slip treatment. For the outlet only the vertical component is set to zero since the default discretization weakly imposes a zero pressure condition if the normal velocity component is not defined.

MATC statements that start with 'MATC' will be evaluated repeatedly while the simulation is running. For more details on using MATC, refer to 'GetStartedElmer.pdf' that contains the old Elmerfem wiki explanation.

Model

```
BoundaryCondition
  Name = Inlet
  Navier-Stokes
    Velocity 1 = Variable Coordinate 2; Real MATC "4*1.5*tx*(0.41-tx)/0.41^2"
    Velocity 2 = 0.0
  Add
  New

  Name = Walls
  Navier-Stokes
    Velocity 1 = 0.0
    Velocity 2 = 0.0
  Add
  New

  Name = Outlet
  Navier-Stokes
    Velocity 2 = 0.0
  Add
  Ok
```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

Model

```
Set boundary properties
  Choose inlet -> set boundary condition Inlet
  Choose both horizontal walls and circle -> set boundary condition Walls
  Choose outlet -> set boundary condition Outlet
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

Sif

```
Generate
Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case.

File

```
Save Project
```

After we have successfully saved the files we may start the solver

Run

```
Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. The norm after the first time step should be around 0.695, and after last 0.749, respectively.

When there are some results to view we may start the postprocessor also

Run

```
Start ParaView
```

Results

Due to the number of the time steps the simulation will take a few minutes. You may inspect the results with Paraview as the time-steps are computed, or wait until all time steps have been computed. You need to reload the files if the number of time steps are increased.

In Figure 23.2 the velocity field is presented for three different time steps. The maximum velocity in the system should be about 2.1724 m/s. Note that here visualization was not performed with Paraview and may therefore look quite different.

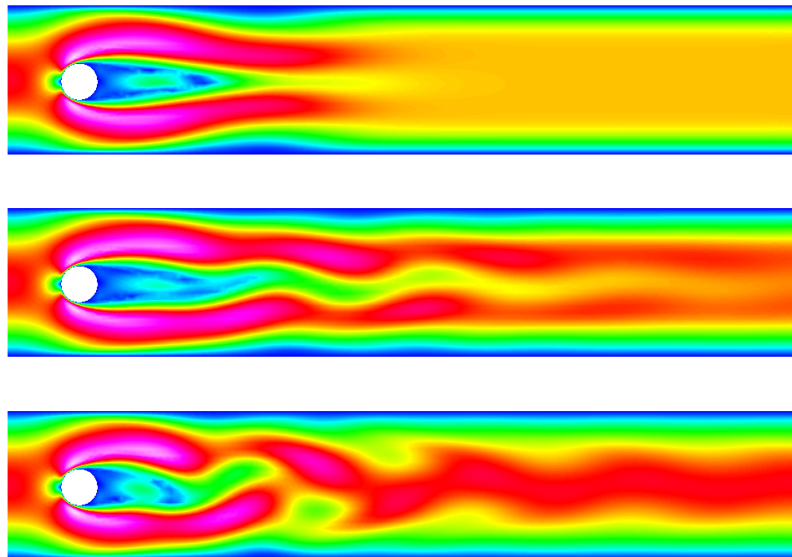


Figure 23.2: Velocity distribution at steps 20, 100 and 200

Effect of Reynolds number

The Reynolds number in this case is around 100 resulting to unsteady flow. The critical Reynolds number is around 90 and reducing the flow velocity so that Reynolds number becomes, say 20, makes the system to possess a steady-state solution. On the other hand, increasing the velocity will make the von Karman vortices even more pronounced until they break into fully chaotic motion. This specific finite element mesh will allow only a minor increase in Reynolds number to be able to capture the phenomena.

Tutorial 24

Thermal flow in curved pipe

Directory: CurvedPipeGUI

Solvers: HeatSolve, FlowSolve

Tools: ElmerGUI

Dimensions: 3D, Steady-state

Author: Thomas Zwinger, Peter Råback

Case definition

This tutorial demonstrates how to set up a coupled case of thermal flow in curved pipe with a finite thickness. Within the pipe both the flow and heat transfer equations need to be solved while on the solid section only heat transfer needs to be considered.

The inner diameter of the pipe is 0.01 m and the outer 0.012 m, respectively. It is bent to a 135 degree angle with a radius of 0.02 m. Beyond the bend 0.03 m of the direct part is also accounted for. The fluid flowing in the pipe is water with an original temperature of 350 K. The outer temperature of the iron pipe is 300 K making the water gradually to cool.

The water is injected with a parabolic velocity profile with a maximum of 0.01 m/s. In reality the laminar analytic profile is described by the Bessel's function. Here the flow is treated as laminar and steady-state even though at these Reynolds number (about 100) the unsteady nature of the flow should probably be considered. This would enhance the heat transfer. The steady-state case, however, will achieve the educational goals of the tutorial.

Solution procedure

The mesh is defined in ElmerGrid format in file `curved_pipe.grd`, load this file.

File

```
Open -> curved_pipe.grd
```

You should obtain your mesh and may check that it consists of 23670 trilinear bricks and 25245 nodes. The density of the mesh may be varied by altering the `Reference Density` in the file. For further information on the mesh definition look at the `ElmerGrid` manual. Often it is desirable to use some professional mesh generation tool in CFD and translate it into Elmer format. For educational purposes we are quite happy to use this simple geometry.

After we have the mesh we start to go through the `Model` menu from the top to bottom. In the `Setup` we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried out in 3-dimensional Cartesian coordinates in steady-state. There is nothing really to be done here, but you may verify that the defaults are correct.

Model

Setup

```
Coordinate system = Cartesian
Simulation type = Steady state
Steady state max. iter = 1
...
```

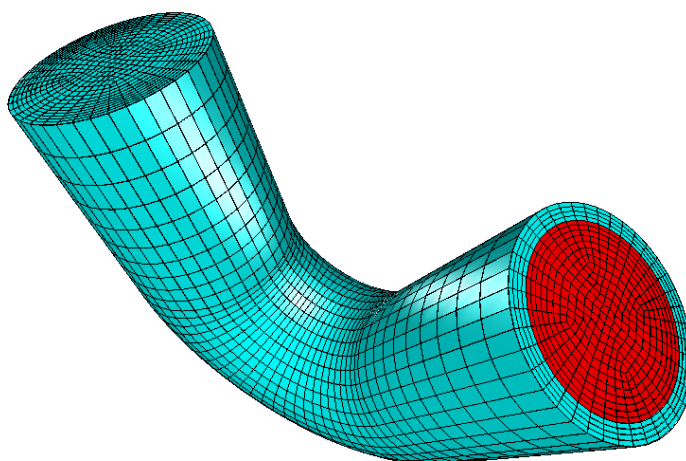


Figure 24.1: The mesh of the curved pipe as seen in ElmerGUI

In the Equation section we choose the relevant equations and parameters related to their solution. In this case we'll have two different sets of solvers (called as Equation in Elmer slang). The other consists of heat and flow solvers, while the other includes just the heat solver. We'll name them appropriately.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we know that the fluid body has the index 1 and the solid body has the index 2. Therefore it is easy to assign the Equation and Material to the bodies directly.

Here we neglect the effect of natural convection. Therefore there is just one-directional coupling from the flow to heat transfer. In order to follow the direction of causality we address the flow solver with a higher priority than the heat solver (default is zero).

Here we are quite happy with the default solver settings of the individual equations. However, for the flow solver we change the default preconditioner `ILU0` to `ILU1` to enhance convergence (with increased memory consumption). For 3D cases the direct solvers are usually not feasible so it is better to stick with the iterative `BiCGstab` linear solver. The equation for the fluid

```
Model
  Equation
    Add
    Name = Heat and Flow
    Apply to Bodies = 1
    Heat Equation
      Active = on
      Convection = Computed
    Navier-Stokes
      Active = on
      Priority = 1
      Edit Solver Setting
        Linear System
          Preconditioning = ILU1
    OK
```

and then for the solid

```
Model
  Equation
    Add
    Name = Just Heat
    Apply to Bodies = 2
    Heat Equation
      Active = on
```

```
Convection = None
OK
```

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the density. Other properties assume a physical law, such as conductivities and viscosity.

Here we choose water and iron from the material library. You may click through the material parameters of the various solvers to ensure that the properties are indeed as they should be. Any consistent set of units may be used in Elmer. The natural choice is of course to perform the computations in SI units.

```
Model
Material
Add
Material library
Water (room temperature)
Apply to Bodies = 1
OK

Add
Material library
Iron (generic)
Apply to Bodies = 2
OK
```

The Body force section usually represents the right-hand-side of an equation. It could be used to account for the natural convection, for example. In this case, however, we do not apply any body forces.

Also an Initial condition could be given in steady-state case to enhance convergence. However, in this case convergence is pretty robust with the default guess of zero.

We have four different boundary conditions: thermal inflow, internal no-slip, outflow, and external fixed temperature. Otherwise natural BCs are assumed. As it is tedious to know the indexes by heart we first define the different BCs and only afterwards apply them to the existing boundaries with the mouse.

```
Model
BoundaryCondition
Name = HotInflow
Heat Equation
Temperature = 350.0
Navier-Stokes
Velocity 1 = 0.0
Velocity 2 = 0.0
Velocity 3 = Variable Coordinate
Real MATC "100.0*(1.0e-4-tx(0)^2-tx(1)^2)"
Add
New
```

The condition for Velocity 3 above may easiest be typed by pressing Enter-key in the edit box which will open a larger window for editing.

```
Name = Outflow
Navier-Stokes
Use normal-tangential coordinate system = on
Velocity 2 = 0.0
Velocity 3 = 0.0
Add
New

Name = NoSlip
Navier-Stokes
NoSlip Wall BC = on
Add
New
```

```
Name = Troom
Heat Equation
  Temperature = 300.0
Add
```

When choosing the boundaries it is important that you choose the right inlet. For that purpose you may activate the compass,

```
View
  Compass = on
```

Now the inlet is the one with normal pointing at the z -direction. Now we are ready to choose the boundaries

```
Model
  Set boundary properties
    Choose inlet face -> set boundary condition HotInflow
    Choose outlet face -> set boundary condition Outflow
    Choose outer side -> set boundary condition Troom
```

Unfortunately we cannot see the internal boundary. For that purpose click on the outer boundary and choose

```
View
  Hide/show selected
```

The outer boundary should vanish and you can proceed with the last BC,

```
Model
  Set boundary properties
    Choose internal side -> set boundary condition Noslip
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
  Generate
  Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case. It's a good idea to give the project an illuminating name. Avoid paths which includes empty spaces since they may cause problems later on.

```
File
  Save Project
    Make New Folder -> curved_pipe
  OK
```

After we have successfully saved the files we may start the solver

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. The simulation may take a few minutes depending on your platform. After the simulation has terminated we may study the results.

Results

The computed norms should be 3.255 for the Navier-Stokes equation and 324.66 for the heat equation. If there is some discrepancy the setup of the system was probably different from the one in the tutorial.

After the simulation has terminated we may open the postprocessor to view the results.

```
Run
  Start ParaView
```

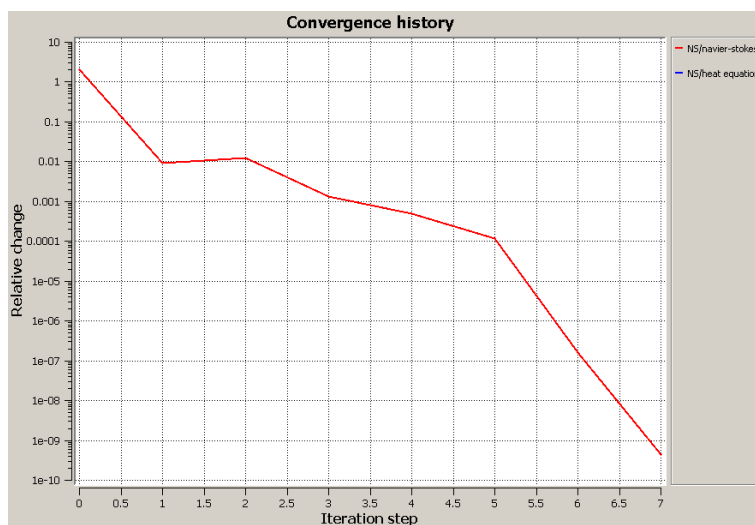


Figure 24.2: Convergence history of the case

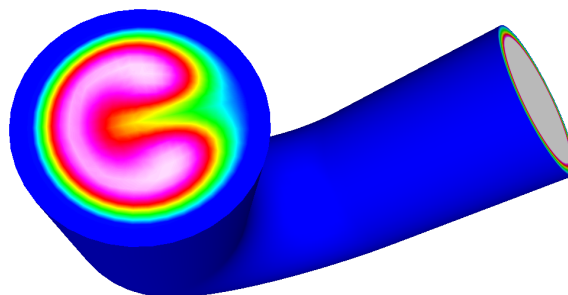


Figure 24.3: Temperature distribution at the outlet of the pipe

A standard way of visualizing is to choose `ColorMesh` and there choose `Surface` and the desired field variable, for example `Velocity_abs` or `Temperature`. In this case only the outflow cross section contains any information. It may be seen in Figure 24.3 that the symmetry around pipe origin is lost in the bend.

Alternatively we may visualize the cross section at $y = 0$. To that aim choose `Isocontours` and there set the `Number Of Isosurfaces` to 1, choose `Surface`, set `Contour Variable` to `nodes_y`, and `Color Variable` to `Temperature` etc. Now you may nicely see how the velocity profile affects the temperature distribution in the pipe.

In Figures 24.5 and 24.4 the obtained velocity and temperature distributions are presented.

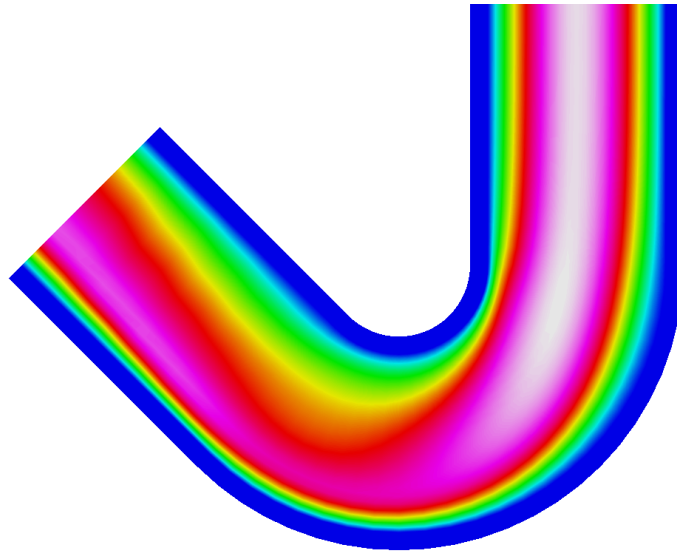


Figure 24.4: Velocity distribution at the cross section $y = 0$.

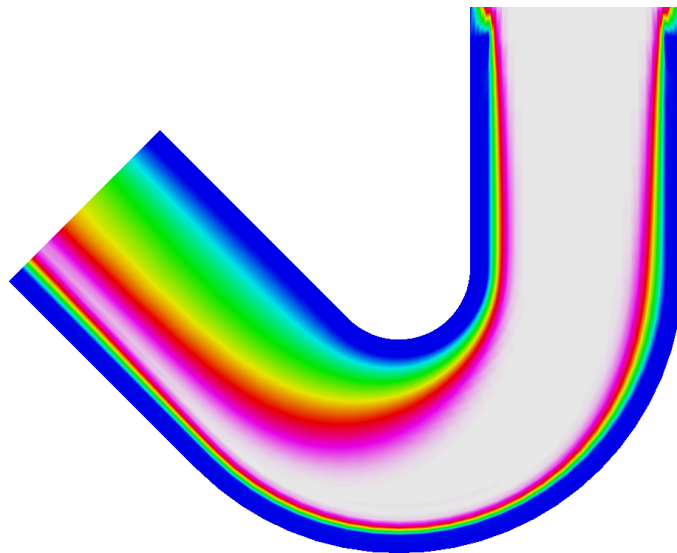


Figure 24.5: Temperature distribution at the cross section $y = 0$.

Tutorial 25

Interaction between fluid flow and elastic obstacle

Directory: FsiObstacleGUI

Solvers: FlowSolve,ElasticSolve,MeshSolve

Tools: ElmerGUI

Dimensions: 2D, Steady-state

Author: Peter Råback

Case definition

This tutorial demonstrates how to set up a coupled case of fluid-structure interaction. Flow is initiated at one end of a channel which has an elastic obstacle that bends under the influence of fluid forces. The resulting displacements modify the domain thereby affecting the flow of the fluid.

The channel is assumed to be 2D. The length of the channel is 10 m and the height is 2 m. At the distance of 2 m from the inlet sits an elastic beam with a rounded top the height of which is 1.2 m and width 0.4 m. A parabolic velocity profile with maximum velocity of 1 m/s is assumed.

Material properties are assumed to be rather simple: For the structure the density is 1000 kg/m^3 , Young's modulus is 1000 Pa, and Poisson ratio 0.3. For the fluid the density is 1 kg/m^3 and viscosity is 0.1 Pas. Additionally the fluid has elastic properties that are used to extend the displacement of the elastic beam to the fluid.

The idea of the case definition is to demonstrate simple yet strongly coupled FSI case without getting into turbulence modelling. Realistic material parameters for the given size of the domain would easily result to turbulence and just small displacements.

The case is solved using standard weak coupling with some relaxation to boost the convergence. The solution is steady-state so only the final results are will be studied.

Solution procedure

The non-linear elasticity equation is not by default active in the menu structures ElmerGUI. Hence, the user must load these before starting the simulations.

```
File
  Definitions
    Append -> nonlinearelastity.xml
```

The additional definitions should reside in the directory `edf-extra` within the distribution. Moving the desired `xml` files to the `edf`-directory enables automatic loading of the definitions at start-up. By inspecting the definitions in the Elmer Definitions File editor one may inspect that the new definitions were really appended.

The mesh is defined in `.in2d` format, the 2D format of netgen, in file `obstacle_in_channel.in2d`, load this file.

```
File
  Open -> obstacle_in_channel.in2d
```

The default mesh is obviously too sparse. To make the mesh more dense set

```
Mesh -> Configure -> nglib -> Max H: 0.1
```

and choose

```
Mesh -> Remesh
```

You should obtain a denser mesh and may check `Model Summary...` that it consists of around 4140 nodes and 7890 linear triangles.

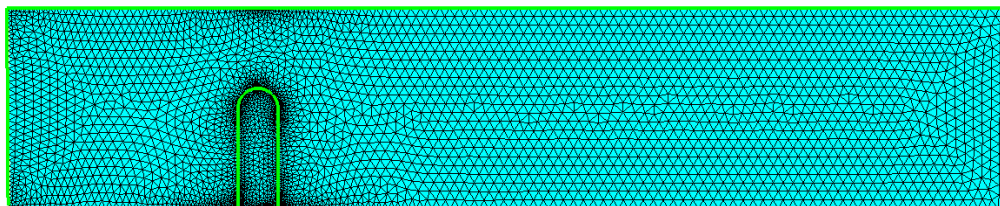


Figure 25.1: The mesh of the obstacle in channel case as seen in ElmerGUI

After we have the mesh we start to go through the `Model` menu from the top to bottom. In the `Setup` we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried out in 2-dimensional Cartesian coordinates in steady-state. There is not much to do here, just increase the number of iterations needed for the convergence of the coupled system. We also set the output interval to zero which means that results are written only at the end of the case.

```
Model
  Setup
    Coordinate system = Cartesian
    Simulation type = Steady state
    Steady state max. iter = 100
    Output Intervals = 0
    ...
```

In the `Equation` section we choose the relevant equations and parameters related to their solution. In this case we'll have two different sets of solvers (called as `Equation` in Elmer slang). The fluid domain consists of flow and mesh deformation solvers, while the elastic domain just includes the non-linear elasticity solver. We'll name them appropriately.

To enhance the convergence and efficient use of resources we set relaxation factors of the primary solvers to 0.5 and the number of non-linear iterations to 1. The mesh deformation solver just extends the displacements of the elasticity solver to the fluid domain, so no relaxation is needed here. For the linear systems we are quite happy with the defaults.

To honor the causality the flow solver should be solved first, then the elasticity solver and at last the mesh deformation solver. We set the priorities accordingly.

The equation for the fluid flow + mesh deformation

```
Model
  Equation
    Add
      Name = Flow and mesh deform
      Apply to Bodies = 1
      Navier-Stokes
        Active = on
        Priority = 2
      Edit Solver Setting
        Nonlinear System
          Max.iterations = 1
          Nonlinear System Relaxation Factor = 0.5
```

```

Mesh Update
  Active = on
  Priority = 0
OK

```

and then for the solid

```

Model
Equation
  Add
  Name = Elasticity
  Apply to Bodies = 2
  Nonlinear Elasticity
  Active = on
  Priority = 1
  Edit Solver Setting
  Nonlinear System
  Max.iterations = 1
  Nonlinear System Relaxation Factor = 0.5
OK

```

Next we set our rather simple material parameters. The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the density. Other properties assume a physical law, such as conductivities and viscosity.

```

Model
Material
  Add
  Name = Ideal fluid
  General
  Density = 1.0
  Navier-Stokes
  Viscosity = 0.1
  Mesh Update
  Elastic Modulus = 1.0
  Poisson Ratio = 0.3
  Apply to Bodies = 1
OK

  Add
  Name = Ideal structure
  General
  Density = 1.0e3
  Nonlinear Elasticity
  Youngs Modulus = 1.0e3
  Poisson Ratio = 0.3
  Apply to Bodies = 2
OK

```

The Body force section usually represents the right-hand-side of an equation. In this case we do not need any body forces.

Also an Initial condition could be given in steady-state case to enhance convergence. However, in this case convergence is pretty robust with the default guess of zero.

We have five different boundary conditions: inflow, outflow, lateral walls with no-slip conditions, fsi conditions, and the beam base. As it is tedious to know the indexes by heart we first define the different BCs and only afterwards apply them to the existing boundaries with the mouse.

```

Model
BoundaryCondition
  Name = Inflow
  Navier-Stokes

```

```

Velocity 1 = Variable Coordinate 2
  Real MATC "tx*(2-tx)"
Velocity 2 = 0.0
Mesh Update 1 = 0.0
Add
New

```

The condition for Velocity 1 above may easiest be typed by pressing Enter-key in the edit box which will open a larger window for editing.

```

Name = Outflow
Navier-Stokes
  Velocity 2 = 0.0
Mesh Update
  Mesh Update 1 = 0.0
Add
New

```

```

Name = Walls
Navier-Stokes
  NoSlip Wall BC = on
Mesh Update
  Mesh Update 1 = 0.0
  Mesh Update 2 = 0.0
Add
New

```

```

Name = Base
Nonlinear Elasticity
  Displacement 1 = 0.0
  Displacement 2 = 0.0
Add
New

```

The essence of fluid-structure interaction is in the following boundary condition. When the Fsi BC is active the fluid forces are automatically within ElasticSolver. The back coupling to Navier-Stokes is achieved through the change in fluid geometry which is enforced by the conditions for the MeshSolver.

```

Name = FSI
Nonlinear Elasticity
  FSI BC = on
Navier-Stokes
  NoSlip Wall BC = on
Mesh Update 1 = Equals Displacement 1
Mesh Update 2 = Equals Displacement 2

```

Now we are ready to choose the boundaries

Model

```

Set boundary properties
  Choose inlet side -> set boundary condition Inflow
  Choose outlet side -> set boundary condition Outflow
  Choose upper and lower sides (three pieces) -> set boundary condition Walls
  Choose obstacle base -> set boundary condition Base
  Choose interface between fluid and solid (two pieces) -> set boundary condition FSI

```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

Sif

```

Generate
Edit -> look how your command file came out

```

Before we can execute the solver we should save the files in a directory. The project includes all the files needed to restart the case. It's a good idea to give the project an illuminating name. Avoid paths which includes empty spaces since they may cause problems later on.

```
File
  Save Project
    Make New Folder -> fsi_obstacle
    OK
```

After we have successfully saved the files we may start the solver

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. The simulation may take around 10 seconds depending on your platform.

The computed norms should be around 0.514 for the Navier-Stokes solver, 0.108 for the elasticity solver, and 0.0548 for the mesh update solver. These are reached after 18 iterations using the rather strict default settings.

If there is some discrepancy the setup of the system was probably different from the one in the tutorial. If the results are in agreement we are ready to look at the results.

Results

To visualize the results open the postprocessor, in this case ParaView After the simulation has terminated we may open the postprocessor to view the results.

```
Run
  Start ParaView
```

For flow problems we can visualize pressure (or velocity amplitudes) separately with the vector presentation of the velocity field. In Paraview the filter to apply the displacement field to the nodal coordinates is called `WarpByVector`.

The maximum speed in the system is around 2.328 and the maximum displacement 0.2636. Note that for the saved results the displacement and mesh update fields have been merged into one. In Figures 25.2, 25.3, and 25.4 the obtained velocity, pressure and displacement distributions are presented in the deformed mesh. A close up showing the deformed mesh is shown in figure 25.5

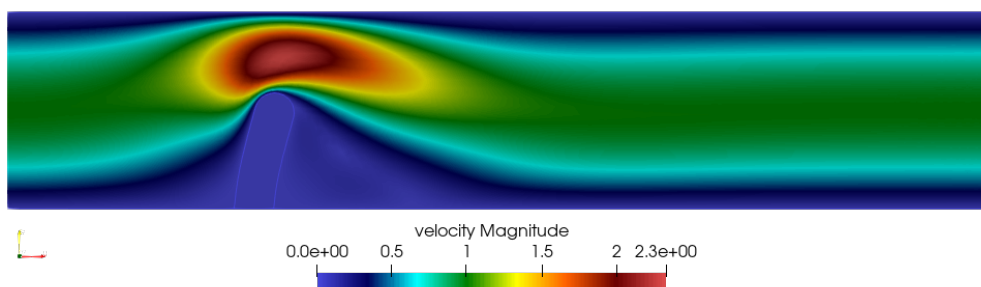


Figure 25.2: Velocity distribution of the obstacle in channel case.

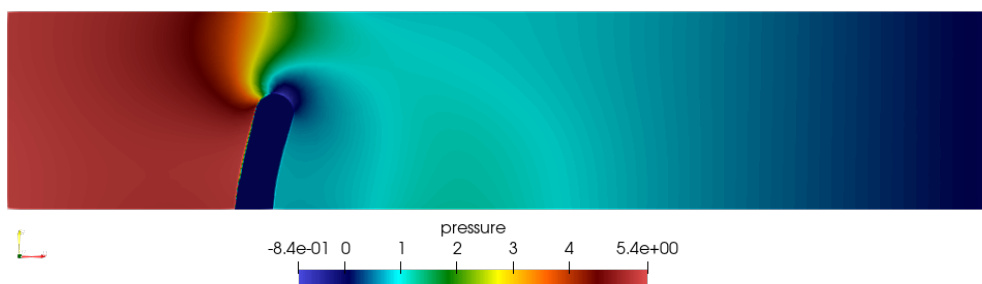


Figure 25.3: Pressure distribution of the obstacle in channel case.

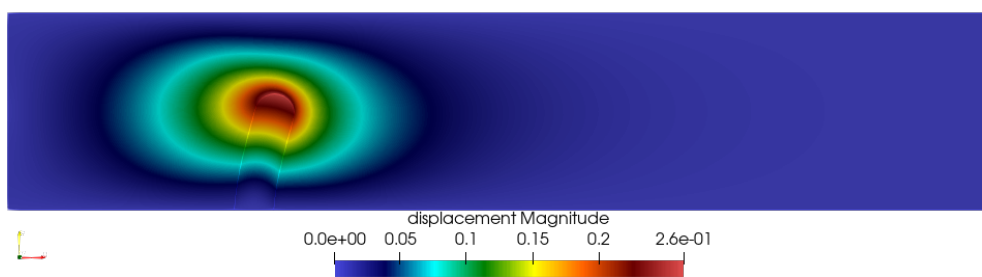


Figure 25.4: Displacement distribution of the obstacle in channel case.

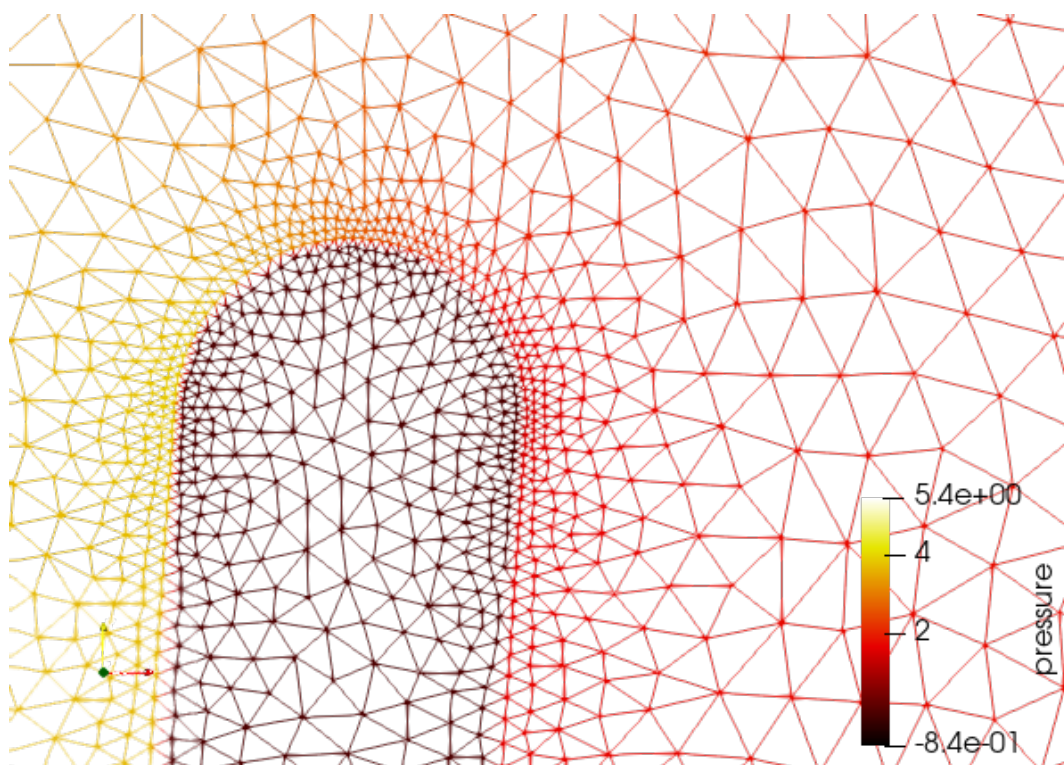


Figure 25.5: Deformed mesh of the obstacle in channel case.

Tutorial 26

Flow and Heat –2D – Transient – Rayleigh-Benard instability

Directory: RayleighBenardGUI

Solvers: HeatSolve, FlowSolve

Tools: ElmerGUI

Dimensions: 2D, Transient

Author: Juha Ruokolainen, Peter Råback

Case definition

This tutorial is about simulating the developing of the Rayleigh-Benard instability in a rectangular domain (Figure 26.1) of dimensions 0.01 m height and 0.06 m length. The simulation is performed with water and the material parameters of water required by the Elmer model are presented in Table 26.1 and can be loaded from the Material Library. The temperature difference between the upper and lower boundary is set to 0.5 so that lower one has the temperature of 293.5 K and the upper one has the temperature of 293 K.

The density of water is inversely proportional to its temperature. Thus, heated water starts to flow upwards, and colder downwards due to gravity. In this case we assume that the Boussinesq approximation is valid for thermal incompressible fluid flow. In other words, the density of the term $\rho \vec{f}$ in the incompressible Navier-Stokes equation can be redefined by the Boussinesq approximation

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

where β is the heat expansion coefficient and the subscript 0 refers to a reference state.

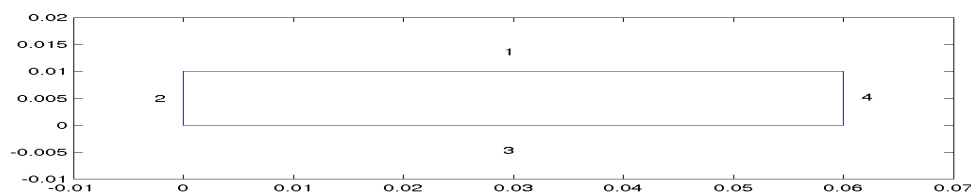


Figure 26.1: Domain.

Table 26.1: Material parameters for water

parameter	value
density	998.3 kg/m ³
viscosity	1040e-6 Ns/m ²
heat capacity	4183 J/(kg·K)
heat conductivity	0.58 W/(m·K)
heat expansion coefficient	2.07e-4 K ⁻¹
reference temperature	293 K

Solution procedure

The mesh is given in ElmerGrid format in file `rectangle.grd`, load this file.

File

```
Open -> rectangle.grd
```

You should obtain your mesh and may check `Model Summary...` that it consists of 3036 bilinear elements. The geometry and mesh should look like figure 26.1.

There is a possibility to divide and unify edges to simplify the case definition in the future.

```
Choose (left wall + right wall (Ctrl down)) -> unify edge
```

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried out in 2-dimensional Cartesian coordinates. 2nd order bdf time stepping method is selected with 200 steps and with step size of two seconds. Gravity is needed for the buoyancy force and it is defined by a vector with four components. The first three components define a unit vector and the fourth its magnitude.

Model

Setup

```
Simulation Type = Transient
Steady state max. iter = 20
Time Stepping Method = bdf
BDF Order = 2
Time Step Intervals = 200
Time Step Sizes = 2.0
Gravity = 0 -1 0 9.82
```

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have one set of equations (named "Natural Convection") which consists of the heat equation and of the Navier-Stokes equation.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and therefore its easier to assign the Equation and Material to it directly. It is important to select the convection to be computed since that couples the velocity field to the heat equation.

The system may include non-linear iterations of each equation and steady state iterations to obtain convergence of the coupled system. It is often a good idea to keep the number of non-linear iterations in a coupled case low. Here we select just one non-linear iteration for both equations.

For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1,...) or direct Umfpack solver, for example.

Model

Equation

```
Name = Natural Convection
Apply to Bodies = 1
Heat Equation
Active = on
Convection = Computed
Edit Solver Setting
```

```

    Nonlinear System
      Max. iterations = 1
  Navier-Stokes
    Active = on
    Edit Solver Setting
      Nonlinear System
        Max. iterations = 1
  Add
  OK

```

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such as conductivities and viscosity.

Here we choose water at room temperature from the Material Library. You may click through the material parameters of the various solvers to ensure that the properties are indeed as they should be. Any consistent set of units may be used in Elmer. The natural choice is of course to perform the computations in SI units.

Apart from the properties from the material database, we enter a reference temperature for the Boussinesq approximation.

```

Model
  Material
    Apply to Bodies = 1
    Material library
      Water (room temperature)
    General
      Reference Temperature = 293
  Add
  OK

```

A Body Force represents the right-hand-side of a equation. It is generally not a required field for a body. In this case, however, we apply the buoyancy resulting from heat expansion as a body force to the Navier-Stokes equation.

```

Model
  Body Force
    Name = Buoyancy
    Apply to Bodies = 1
    Navier-Stokes
      Boussinesq = on
  Add
  OK

```

Initial conditions should be given to transient cases. In this case we choose a constant Temperature field and an small initial velocity that initializes the symmetry break.

```

Model
  Initial Condition
    Name = Initial Guess
    Heat Equation
      Temperature = 293
    Navier-Stokes
      Velocity 1 = 1.0e-9
      Velocity 2 = 0.0

```

Only one boundary condition may be applied to each boundary and therefore all the different physical BCs for a boundary should be grouped together. In this case the Temperature and Velocity. The side walls are assumed to be adiabatic.

```

Model
  BoundaryCondition
    Name = Bottom
    Heat Equation

```

```

    Temperature = 293.5
Navier-Stokes
    Velocity 1 = 0.0
    Velocity 2 = 0.0
Add
New

Name = Top
Heat Equation
    Temperature = 293
Navier-Stokes
    Velocity 1 = 0.0
    Velocity 2 = 0.0
Add
New

Name = Sides
Navier-Stokes
    Velocity 1 = 0.0
    Velocity 2 = 0.0
Add

```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

```

Model
Set boundary properties
    Choose Bottom -> set boundary condition Bottom
    Choose Top -> set boundary condition Top
    Choose Sides -> set boundary condition Sides

```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```

Sif
Generate
Edit -> look how your command file came out

```

Before we can execute the solver we should save the files in a directory. The ElmerGUI project includes all the files needed to restart the case.

```

File
Save Project

```

After we have successfully saved the files we may start the solver

```

Run
Start solver

```

A convergence view automatically pops up showing relative changes of each iteration.

When there are some results to view we may start the postprocessor also

```

Run
Start ParaView

```

Results

Due to the number of the time steps the simulation may take around ten minutes. You may inspect the results with Paraview as the time steps are computed, or wait until all time steps have been computed. You must reload the files if their number has changed. The time series can be automatically animated and even saved to an animation file.

In Figures 26.2 through 26.3 the obtained temperature and velocity distribution are presented. The maximum velocity in the system should be about 0.516 mm/s.

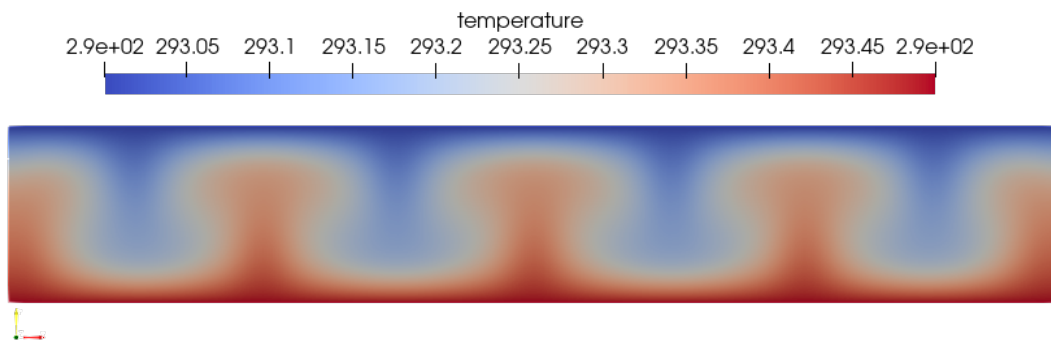


Figure 26.2: Temperature distribution

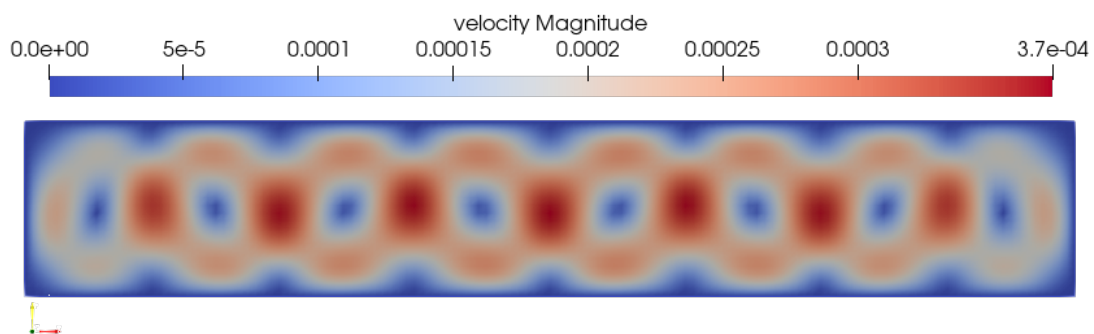


Figure 26.3: Velocity magnitudes

Extra task: Sensitivity to temperature difference

If you have time you may try to solve the case with different parameters. Changing the temperature difference is one way of affecting the instability of the system. Decreasing the temperature differences the system eventually becomes steady state and the convection rolls vanish altogether. Increasing the temperature difference may increase the number of convection rolls and eventually the system becomes fully chaotic. Note that changing the temperature difference also affects to the time scale of the wake.

Tutorial 27

Temperature distribution of a toy glacier

Directory: ToyGlacierTemperature

Solvers: HeatSolver

Tools: ElmerGUI,nglib

Dimensions: 2D, Steady-state

Author: Peter Råback, Thomas Zwinger

Introduction

The purpose of this simple tutorial is to be an introduction into Elmer for people dealing with computational glaciology. This tutorial shows how to apply one equation and related boundary conditions to just one domain.

Problem description

Consider a 2D toy model of a glacier with length of 7000 m and thickness of about 1000 m. There is a slight declination in the geometry which will make the glacier flow to the left. The left-hand-side is rounded to imitate a true glacier while the right-hand-side is cut off.

We solve for the temperature distribution T of the glacier. A heat flux of $q = 0.02 \text{ W/m}^2$ is applied at the bottom of the glacier while the surface stays at a fixed temperature of $T_0 = -10 \text{ C}$. The material properties of ice are used for the heat conductivity $\kappa(T)$. The temperature distribution in the glacier may be solved from

$$\begin{cases} -\kappa\Delta T & = & 0 & \text{in } \Omega \\ T & = & T_0 & \text{on } \Gamma_D \\ \kappa\frac{\partial T}{\partial n} & = & q & \text{on } \Gamma_N \end{cases} \quad (27.1)$$

Starting and meshing

Start ElmerGUI from command line or by clicking the icon in your desktop (or in the /bin directory of your installation). Here we describe the essential steps in the ElmerGUI by writing out the clicking procedure. Indentation generally means that the selections are done within the window chosen at the higher level.

The mesh is given in ElmerGrid format in file `glacier_toy.in2d` in the samples directory of ElmerGUI, load this file.

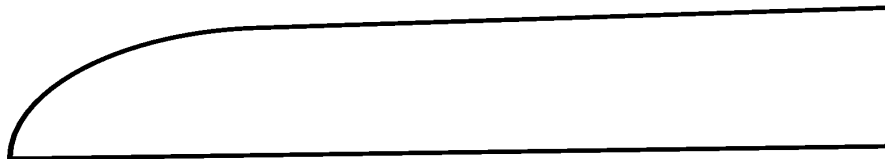


Figure 27.1: The shape of the toy glacier to be studied

File

```
Open -> glacier_toy.in2d
```

You should obtain a mesh consisting of just two triangular elements. The mesh is created by the `netgen` plugin and in order to increase the mesh density the in-line parameters of `netgen` must be defined in ElmerGUI. Here we set the maximum element size to 50.

Mesh

```
Configure...
  nglib -> Max H: 50
```

The resulting mesh should consist of 3335 nodes and 6355 triangles as may be checked in the `Model` summary window.

Command file definition

After we have the mesh we start to go through the `Model` menu from the top to bottom. In the `Setup` we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried out in 2-dimensional Cartesian coordinates and in steady-state. Only one steady-state iteration is needed as the case is linear.

Model

```
Setup
  Simulation Type = Steady state
  Steady state max. iter = 1
```

Choose `Apply` to close the window.

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have one set only one equation – the heat equation.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and one boundary and therefore its easier to assign the Equation and Material to it directly.

For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1,...) or direct Umfpack solver, for example.

Model

```
Equation
  Add
    Name = Heat Equation
    Apply to bodies = 1
    Heat Equation
    Active = on
  Apply
  OK
```

The `Material` section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such heat conductivity. We choose ice from the `Material` library which automatically sets for the needed material properties.

Model

```
Material
  Add
    Material library
    Water (frozen)
    Apply to bodies = Body 1
  Add
  OK
```

This includes, for example, temperature dependent heat conductivity as may be seen under the `HeatEquation` page of the material properties. `MATC` language is used here to define the functional form.

A `Body Force` represents the right-hand-side of a equation that in this case represents the heat source. In this case there are no internal heat sources so we do not need one. Also no `Initial Condition` is required in steady state case.

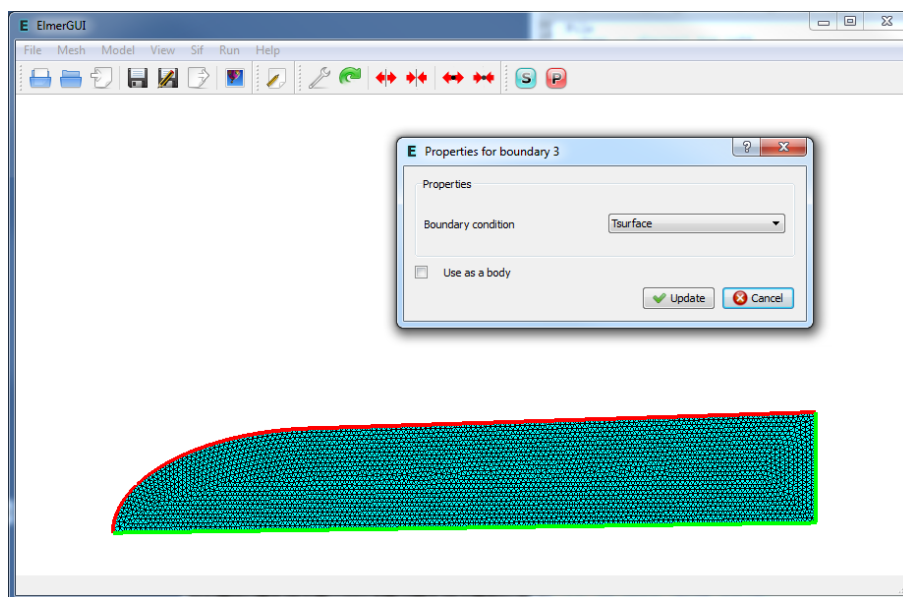


Figure 27.2: Defining boundary conditions in ElmerGUI session

We set three different kinds of boundary conditions. A fixed temperature, a fixed flux and natural boundary condition (zero flux). As there are several boundaries we first define the different boundary types, and thereafter assign them using the mouse. A screenshot of the case when setting the BCs is shown in figure 27.2.

Model

```

BoundaryCondition
  Add
    Name = Tsurface
    Heat Equation
      Temperature = -10.0
    OK
  Add
    Name = Tflux
    Heat Equation
      Heat Flux = 0.02
    OK
  Add
    Name = Tnatural
    OK

```

Then we set the boundary properties

Model

```
Set boundary properties
```

Choose the correct boundary by clicking with the mouse and apply the condition for this boundary.

Boundary condition

```

Click top boundary -> choose Tsurface
Click bottom boundary -> choose Tflux
Click r.h.s. boundary -> choose Tnatural

```

Saving and solution

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

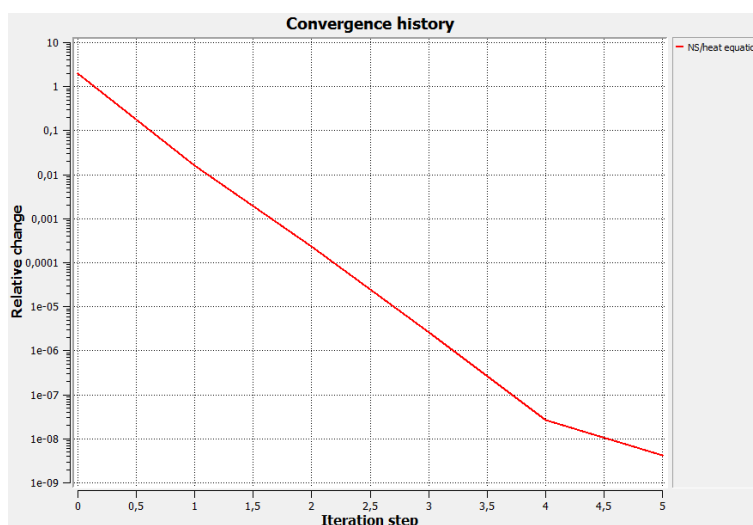


Figure 27.3: The convergence ElmerGUI

```
Sif
  Generate
  Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. In saving the project all the necessary files for restarting the case will be saved to the destination directory.

```
File
  Save Project
```

After we have successfully saved the files we may start the solver

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. The heat conductivity of ice is set to be dependent on temperature and this results to a non-linear iteration. The resulting output is shown in figure 27.3.

Note: if you face problems in the solution phase and need to edit the setting, always remember to save the project before execution.

Results

To view the results we use Paraview,

```
Run
  Start Paraview
```

Picture 27.4 shows the surface mesh coloured with temperature. Note that these results were carried out with the obsolete VTK based tool within ElmerGUI, and therefore look different than in Paraview.

The maximum temperature obtained with the above choices is 0.11166 C. With a denser mesh the result is naturally more accurate, but solving the problem takes more calculation time.

You may study the effect of mesh refinement by choosing a different value for the `Max H` parameter. under `Configure`. After choosing `Remesh` and saving the mesh the solver may be recalled with the modified mesh.

Transient simulation

We use the steady-state simulation presented above as our starting point and solve a transient version of it. Initially the glacier is assumed to be at -10 C and it is gradually heated from below.

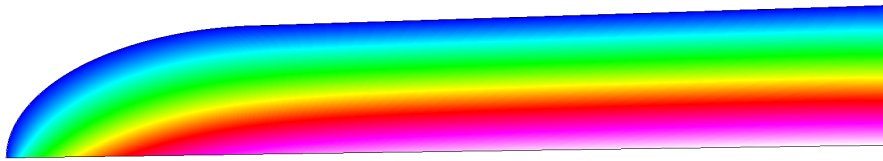


Figure 27.4: The temperature distribution of the toy glacier.

We use 2nd order bdf time stepping method is selected with 100 steps and with step size of 100 years - melting the ice from below with such a small flux would take quite a few years. The mathematical expression followed by “\$” is evaluated when the command file is read. Here it is used to transform the time in years to those in one second.

Model

Setup

```
Simulation Type = Transient
Time Stepping Method = bdf
BDF Order = 2
Time Step Intervals = 100
Time Step Sizes = $ 3600 * 24 * 365.25 * 100
Gravity = ...
```

Initial conditions should be given to transient cases. In this case we choose a constant Temperature of -10 C. This is consistent with the boundary condition at the top wall.

Model

Initial Condition

```
Name = Initial Guess
Heat Equation
Temperature = -10.0
```

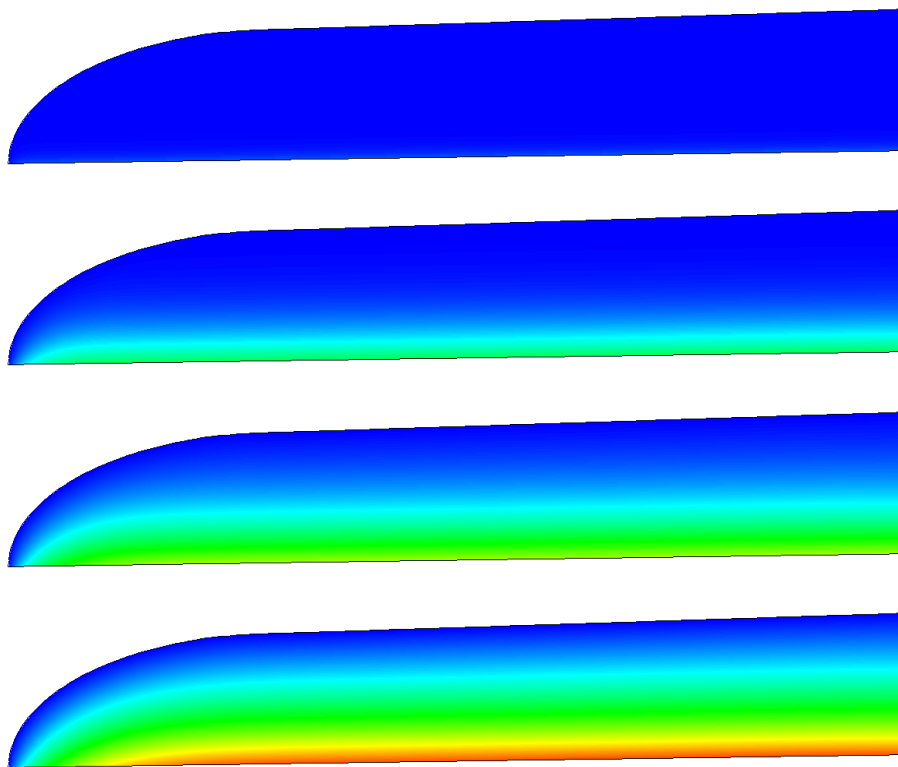


Figure 27.5: Temperature distribution after 1, 20, 50 and 100 time steps. The temperature scale is the same that is used in the steady-state case. The maximum temperature at end should be about -3.7611 C.

Tutorial 28

Temperature and velocity distributions of a toy glacier and bedrock

Directory: ToyGlacierTemperatureAndFlow

Solvers: HeatSolver,FlowSolver

Tools: ElmerGUI,nglib

Dimensions: 2D, Steady-state

Author: Thomas Zwinger, Peter Råback

Introduction

The purpose of this simple tutorial is to be an introduction into Elmer for people dealing with computational glaciology. The tutorial is a continuation of the previous one with slightly more complex geometry. This tutorial shows how to apply two different solvers to two different bodies.

Problem description

Consider a 2D toy model of a glacier sitting on a piece of bedrock. With the bedrock present it is possible to study more accurately the temperature profiles.

Now we solve for the temperature distribution T for the combined system. A heat flux of $q = 0.02 \text{ W/m}^2$ is applied at the bottom of the bedrock while the surface stays at a fixed temperature of $T_0 = -10 \text{ C}$. For ice the properties from the database are assumed while for the bedrock density is assumed to be 2500 kg/m^3 and heat conductivity 3 W/mK .

We also solve for the velocity distribution \vec{v} of the glacier. The velocity field is solved from the Stokes equation and is assumed to be affected only by the Gravity. As boundary conditions we apply a no-slip condition at the ice-rock interface and symmetry condition at the right-hand-side of the glacier.

Starting and meshing

Start ElmerGUI from command line or by clicking the icon in your desktop (or in the /bin directory of your installation). Here we describe the essential steps in the ElmerGUI by writing out the clicking procedure. Tabulation generally means that the selections are done within the window chosen at the higher level.

The mesh is given in ElmerGrid format in file `glacier_on_bedrock_toy.in2d` in the samples directory of ElmerGUI, load this file.

File

```
Open -> glacier_on_bedrock_toy.in2d
```

When netgen is ready with the meshing. You should obtain a mesh consisting of 4329 nodes and 8406 triangular elements. Now the mesh density was predefined in the `in2d` file and therefore no command-line arguments are needed to refine the mesh. If you got more elements check the value of `Max H` in the netgen parameter window.

If the mesh was successfully imported your window should look something in figure 28.1.

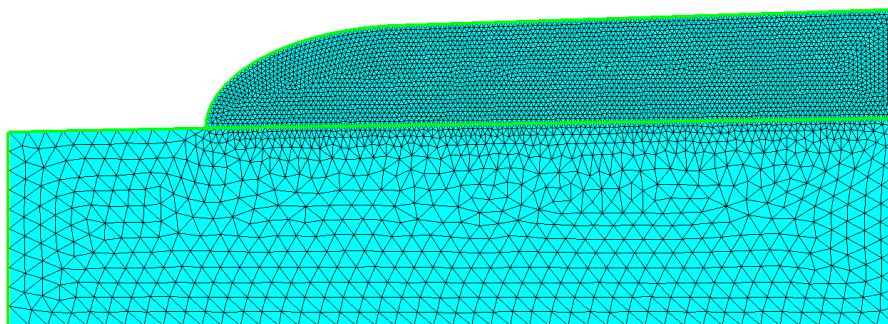


Figure 28.1: The finite element mesh in ElmerGUI

Command file definition

After we have the mesh we start to go through the Model menu from the top to bottom. Again we are happy with the definitions in the Setup window.

In the Equation section we choose the relevant equations and parameters related to their solution. In this case we'll have two different sets of solvers (called as Equation in Elmer slang). The first consists of heat and flow solvers, while the other includes just the heat solver. We'll name them appropriately.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we know that the fluid body has the index 1 and the solid body has the index 2. Therefore it is easy to assign the Equation and Material to the bodies directly.

Here we neglect the effect of convection to the temperature distribution. Therefore there is no coupling from velocity field to energy equation. However, the temperature is affecting the viscosity of ice and therefore it needs to be solved first. This is ensured by giving higher priority to the heat solver (default is zero). In order to obtain the Stokes equation convection of momentum is omitted from the Navier-Stokes equations.

Here we are quite happy with the default solver settings of the individual equations of the linear systems. However, the user may play around with different linear system settings to study their effect on convergence and computation time. The equation for the ice

```
Model
Equation
Add
Name = Heat and Flow
Apply to Bodies = 1
Heat Equation
Active = on
Priority = 1
Convection = None
Navier-Stokes
Active = on
Convect = off
OK
```

and then for the solid

```
Model
Equation
Add
Name = Just Heat
Apply to Bodies = 2
Heat Equation
Active = on
Convection = None
OK
```

The Material section includes the material parameters. We choose ice from the Material library which automatically sets for the needed material properties. For the bedrock we define the two parameters that are required.

```
Model
  Material
    Add
      Material library
        Water (frozen)
        Apply to bodies = Body 1
    Add
      OK
    Add
      Name = Bedrock
      Apply to bodies = Body 2
      General
        Density = 2500.0
      Heat Equation
        Heat Conductivity = 3.0
    Add
      OK
```

A Body Force represents the right-hand-side of a equations. For the heat equation there are no source terms. For the Stokes equation we apply gravity which points to the negative y direction.

```
Model
  BodyForce
    Name = Gravity
    Navier-Stokes
      Force 2 = -9.81
    Apply to Bodies = Body 1
    Add
      OK
```

We do not need any Initial Condition since the zero temperature (in Celsius) is a good initial guess for the heat equation. For the Stokes equation a better solution could be used since if convergence problems would arise since the non-Newtonian material laws do actually depend on the initial velocity.

We set four different kinds of boundary conditions. A fixed temperature, a fixed flux, no-slip condition and symmetry condition. As there are several boundaries we first define the different boundary types, and thereafter assign them using the mouse.

```
Model
  BoundaryCondition
    Add
      Heat Equation
        Temperature = -10.0
      Name = Tsurface
    Add
      OK
    Add
      Heat Equation
        Heat Flux = 0.02
      Name = Tflux
    Add
      OK
    Add
      Navier-Stokes
        No-slip Wall BC = on
      Name = NoSlip
    Add
      OK
    Add
      Navier-Stokes
        Velocity 1 = 0.0
      Name = Symmetry
    Add
      OK
```

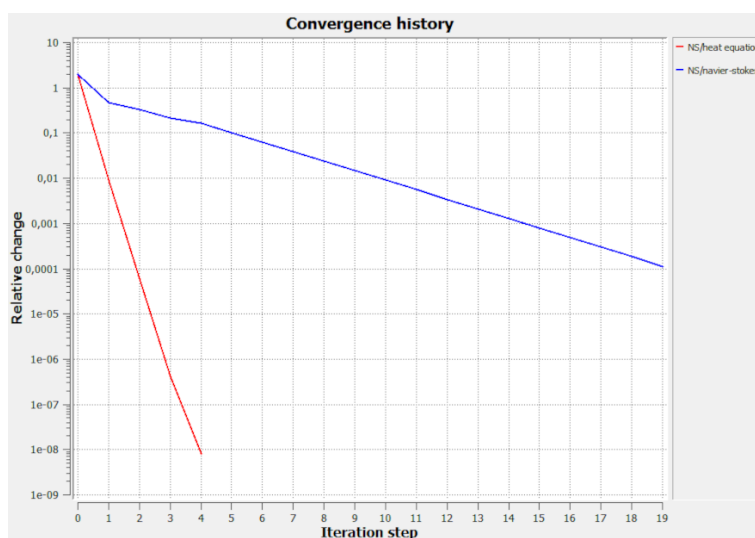


Figure 28.2: The convergence ElmerGUI

Then we set the boundary properties

Model

Set boundary properties

Choose the correct boundary by clicking with the mouse and apply the condition for this boundary.

Boundary condition

Click top boundary of ice -> choose Tsurface

Click the bare part of bedrock -> choose Tsurface

Click bottom boundary of bedrock -> choose Tflux

Click bottom boundary of ice -> choose NoSlip

Click r.h.s. boundary of ice -> choose Symmetry

Saving and solution

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

Sif

Generate

Edit -> look how your command file came out

Before we can execute the solver we should save the files in a directory. In saving the project all the necessary files for restarting the case will be saved to the destination directory.

File

Save Project

After we have successfully saved the files we may start the solver

Run

Start solver

A convergence view automatically pops up showing relative changes of each iteration. The heat conductivity of ice is set to be dependent on temperature and this results to a non-linear iteration. The resulting output is shown in figure 28.2.

Note: if you face problems in the solution phase and need to edit the setting, always remember to save the project before execution.

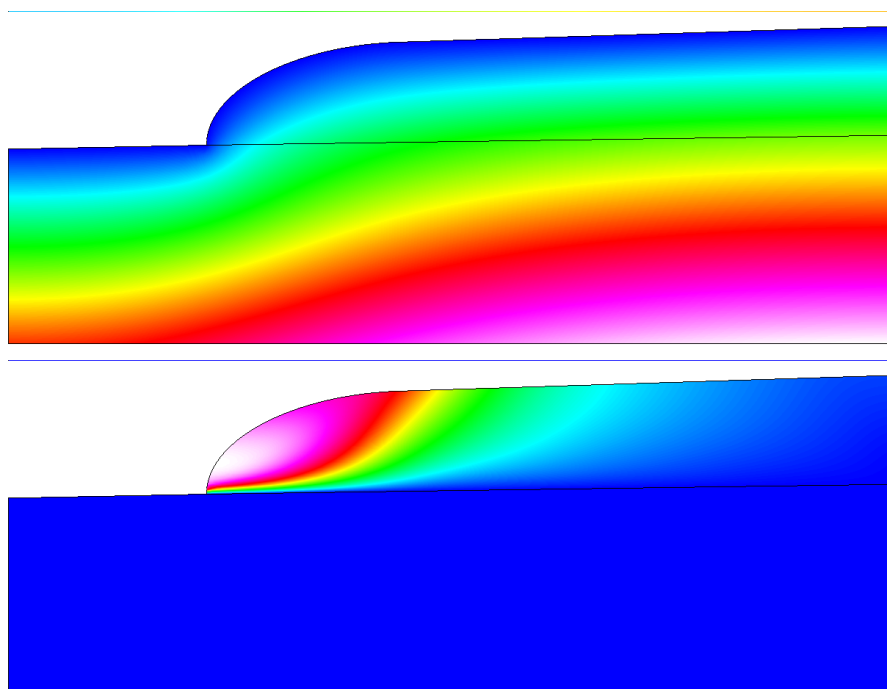


Figure 28.3: Temperature (upper figure) and velocity (lower figure) distributions of the toy glacier sitting on a bedrock.

Results

To view the results we use Paraview,

Run

```
Start Paraview
```

The resulting temperature and velocity distributions are shown in figure 28.3 (these are generated with the obsolete VTKPost tool within ElmerGUI).

The maximum temperature obtained with the above choices is 12.955 C. For the velocity the maximum absolute value is 6.3337 mm/s which is actually unreasonably high since it corresponds to yearly movement of around 200 km. This just shows that the shape of the toy glacier under study is very unrealistic.

Tutorial 29

Generic scalar PDE on 3D angle domain

Directory: ModelPDE3D

Solvers: ModelPDE

Tools: ElmerGUI

Dimensions: 3D, Steady-state

Author: Peter Råback

Problem description

This tutorial demonstrates the use of the generic advection-diffusion-reaction equation through ElmerGUI. The solver may be found in module ModelPDE. The purposes of the model pde and also this tutorial is to help those who want to understand Elmer from a mathematical perspective to be able to carry out some of their own code development.

The problem is a simple 3d structure `winkel.grd` that can be characterized by a $2 \times 2 \times 2$ topological grid where entries $(1, 1, 1)$, $(1, 2, 1)$, $(2, 1, 1)$ and $(2, 1, 2)$ are meshed. This is the simplest Cartesian structure with full 3D solution.

We can rather freely play with the parameters of the Model PDE. The equation is generic and the parameters are assumed to be unit free. For detailed description of the problem see the description in Elmer Programmers Tutorial.

As a first suggestion, we will show how to make a simple case where the two extreme edges are set to zero using Dirichlet boundary conditions, and constant unity source term is applied to the body. This is the simple Poisson equation with constant coefficient.

Menu structures for Model PDE

The menu structures for the case are defined in `model-pde.xml`. If after starting you cannot find the menu structures add the file to the `edf` directory of your installation, or append the menu structures within ElmerGUI.

The following material parameters may be defined

Diffusion Coefficient Real
Diffusion coefficient, μ .

Reaction Coefficient Real
Reaction coefficient, λ .

Time Derivative Coefficient Real
Multiplier of the time derivative, ρ .

Convection Coefficient Real
Multiplier of convection coefficient, κ .

Convection Velocity 1 Real
Convection velocity in direction x , a_x .

Convection Velocity 2 Real
Convection velocity in direction y , a_y .

Convection Velocity 3 Real
Convection velocity in direction z , a_z .

The following parameter defines the heat source f on the right-hand-side

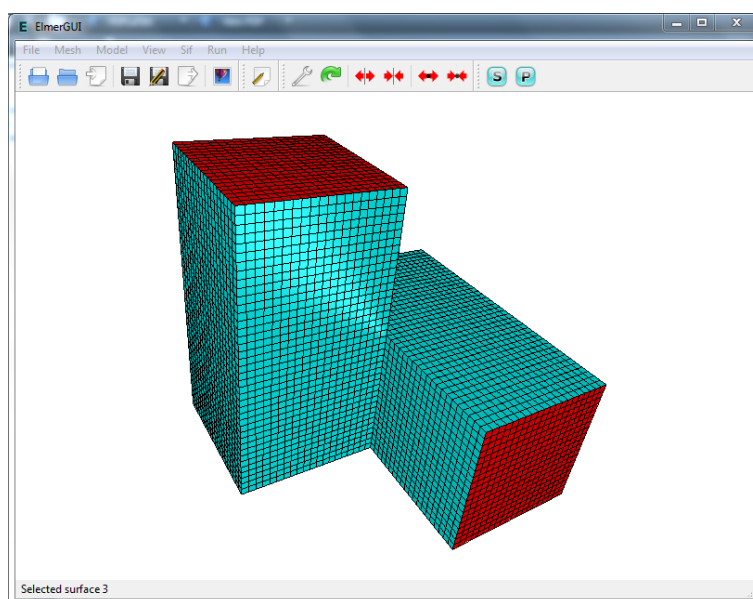


Figure 29.1: The finite element mesh in ElmerGUI

```
Field Source Real
```

The menu structures defines the following parameters for boundary conditions:

```
Field Real
```

Dirichlet BC for the scalar field under study, u .

```
Field Flux Real
```

Neumann boundary condition for the field, q .

```
Robin Coefficient Real
```

```
External Field Real
```

Coefficient α and external field value g for Robin boundary condition.

In transient cases the user may also give an initial condition for u ,

```
Field Real
```

Solution procedure

Start ElmerGUI from command line or by clicking the icon in your desktop. Here we describe the essential steps in the ElmerGUI by writing out the clicking procedure. Indentation generally means that the selections are done within the window chosen at the higher level.

The mesh is given in ElmerGrid format in file `winkel.grd` in the samples directory of ElmerGUI, load this file.

```
File
```

```
Open -> winkel.grd
```

You should obtain your mesh and may check in the Model summary window that it consists of 35 721 nodes and 32 000 trilinear elements. If the mesh was successfully imported your window should look something in figure 29.1. The figure also shows the two extreme boundary patches that we intend to use Dirichlet conditions for.

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried out in 2-dimensional Cartesian coordinates and in steady-state. Only one steady-state iteration is needed as the case is linear.

```
Model
```

```
Setup
```

```
Simulation Type = Steady state
```

```
Steady state max. iter = 1
```

Choose Apply to close the window.

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have one set only one equation – the Model PDE.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and one boundary and therefore its easier to assign the Equation and Material to it directly.

For the linear system solvers we are happy to use the defaults. One may however, try out different preconditioners (ILU1,...) or direct Umfpack solver, for example.

```
Model
  Equation
    Add
      Name = Model PDE
      Apply to bodies = 1
      Model PDE
        Active = on
    Add
  OK
```

The Material section includes all the material parameters. If material parameter is not defined. It is assumed to be zero. Here we just set the diffusivity to one.

```
Model
  Material
    Name = Ideal
    Apply to bodies = 1
    Model PDE
      Diffusion Coefficient = 1.0
    Add
  OK
```

A Body Force represents the right-hand-side of a equation,

```
Model
  Body Force
    Name = Source
    Apply to bodies = 1
    Model PDE
      Field Source = 1.0
    Add
  OK
```

No initial conditions are required in steady state case.

Finally, for the BCs first define them and then use the mouse to apply them to the correct boundary patches. Refer to figure 29.1 and select the two surfaces as shown.

```
Model
  BoundaryCondition
    Name = Zero
    Model PDE
      Field = 0.0
    Add
  OK
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
  Generate
  Edit -> look how your command file came out
```

Before we can execute the solver we should save the files in a directory. In saving the project all the necessary files for restarting the case will be saved to the destination directory.

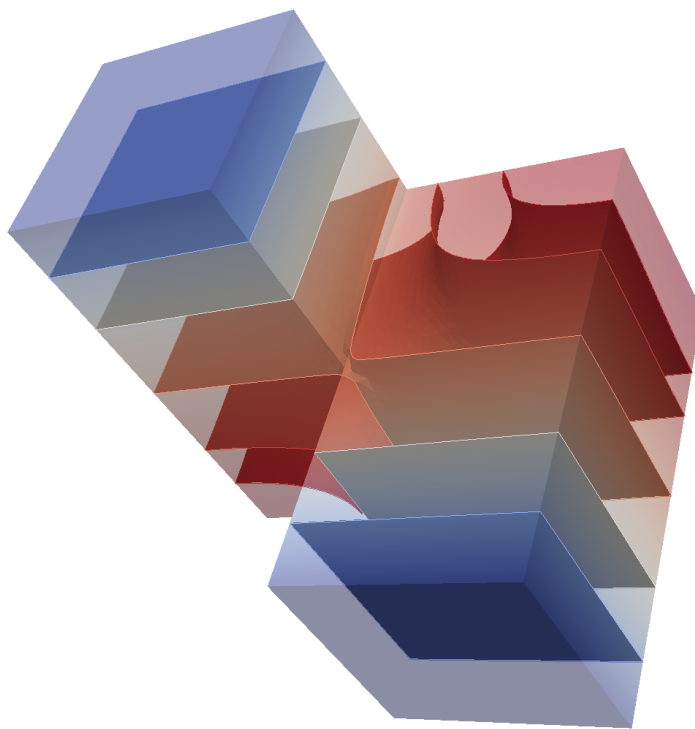


Figure 29.2: The field values of the structure as visualized with Paraview. Isosurfaces are defined for field values 0.5, 1.0, 1.5, 1.8 and 1.9, respectively.

File

Save Project

After we have successfully saved the files we may start the solver

Run

Start solver

A convergence view automatically pops up showing relative changes of each iteration. As the case is linear only one iteration was required for the solution and the second one just is needed to check the convergence.

The norm of the results at convergence should be 1.4243820.

Note: if you face problems in the solution phase and need to edit the setting, always remember to save the project before execution.

To view the results we may visualize them with Paraview.

Further possibilities

Here are some things you could try out, or are at least possible. The menus of the GUI are just elementary so more advanced features may need that the keywords are added by hand to the command file. No coding is needed to implement these features though.

- You can also solve the problem with an unstructured tetrahedral mesh using Gmsh format file `winkel.msh`.
- Play with the reaction, diffusion, convection coefficient, and also the advection velocity. As far as they remain constant the equation should be solvable with one sweep.
- You could try to use Neumann or Robin BCs as well. Remember though that a steady state equation needs definitions that uniquely define the solution.
- Make the problem time-dependent. Note that then you most likely need to define the coefficient for the time derivative.

- When the advection increases in size the solution may eventually become oscillatory. To eliminate that some bubbles or p-elements may be needed. You may play around with element settings, for example use `Element = p:2` or `Element = n:1 b:1` etc.
- You could try to increase the number of elements either by using `-relh` parameter in ElmerGUI, or setting `Mesh Levels = 2` in Simulation section.
- You could try to use `MATC` to make the coefficients parameter dependent. Dependence on the solution itself introduces non-linearities that might not be well handled by the fixed point iteration scheme. For more demanding non-linearities Newton linearisation or other techniques may be needed.
- Also some periodicity could be introduced to this problem by letting the two extreme surface patches have a dependence between them.
- You could introduce sort of contact conditions for the surface or bulk values of the problem by defining minimum or maximum values for the field.

Tutorial 30

Electro-kinetics – 2D – Electro-osmotic flow and advected species

Directory: Electrokinetics

Solvers: StatElecSolve, FlowSolve, AdvectionDiffusion, Electrokinetics

Tools: ElmerGUI

Dimensions: 2D, Transient

Author: Original author not known

Case definition

This tutorial is an example of setting up a simulation for (microfluidic) electro-osmotic flow advecting a passive scalar quantity. Diffusion of the species is also included. The geometry of the system is a simple 2D microchannel with T crossing. The flow is induced by the applied electric field and the electric double layer at the channel walls. The analyte (species) is inserted into the system from the left hand side inlet.

More details on the electro-kinetic capabilities of Elmer are found on the Models Manual, chapter “Electrokinetics”, authored by T. Zwinger.

This is an old case of analyte transport with Helmholtz-Smoluchowski Boundary Conditions. Not perhaps particularly accurate as the advection-diffusion equation suffers from high numerical diffusion. Particularly for purely advective transport the particle based methods might be more favourable.

The geometry for this tutorial looks like as shown in figure 31.1.

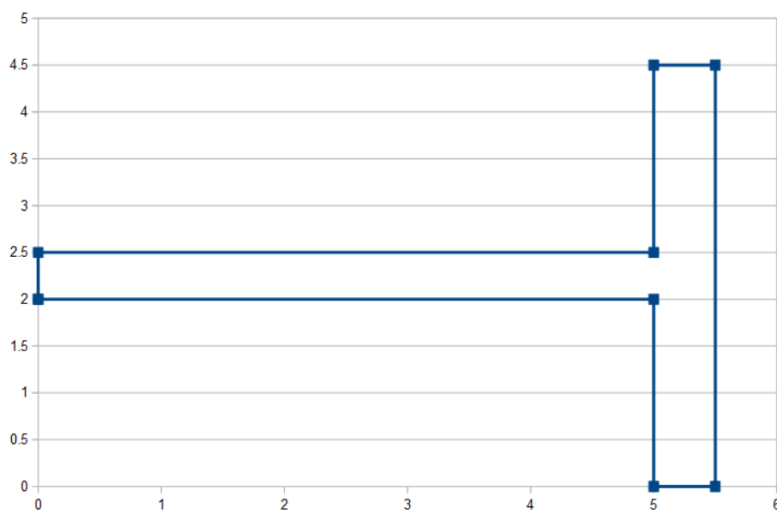


Figure 30.1: Geometry

ElmerGUI Equation Menu

The GUI definitions of Advection Diffusion solver utilized in this tutorial are located in the `edf-extra` folder and thus need to be manually activated in the ElmerGUI.

Note that the extra definition needs to be loaded first, before any other steps in building any tutorial. Once the extra definition has been loaded, and the ElmerGUI project has been saved, then the reference to the extra definition will be stored in the project, and you won't have to load the extra definition again.

```
File
  Definitions
    Append -> advection-diffusion.xml
  Close
```

The `advection-diffusion.xml` definition file is, by default, located in Linux in:

```
$ELMER_HOME/share/ElmerGUI/edf-extra
```

and in Windows, located in:

```
C:/Program Files/Elmer 9.0-Release/share/ElmerGUI/edf-extra
```

We will also be using the Stat Elec Solver, and the Flow Solver which are two of the default, pre-loaded GUI definitions, so they do not need to be manually activated. For reference, the `electrostatics.xml` and the `navier-stokes.xml` definition files are, by default, located in Linux in:

```
$ELMER_HOME/share/ElmerGUI/edf
```

and in Windows, located in:

```
C:/Program Files/Elmer 9.0-Release/share/ElmerGUI/edf
```

The Electrokinetics solver is used for a boundary condition and will be called when needed. The Electrokinetics solver is located in the `$ELMER_HOME/bin` folder.

Solution procedure

The mesh is given in ElmerGrid format in file `Tcross.grd`, load this file.

```
File
  Open -> Tcross.grd
```

You should obtain your mesh and may check `Model Summary...` that it consists of 1200 surface elements. Your mesh should look like as shown in figure 31.5

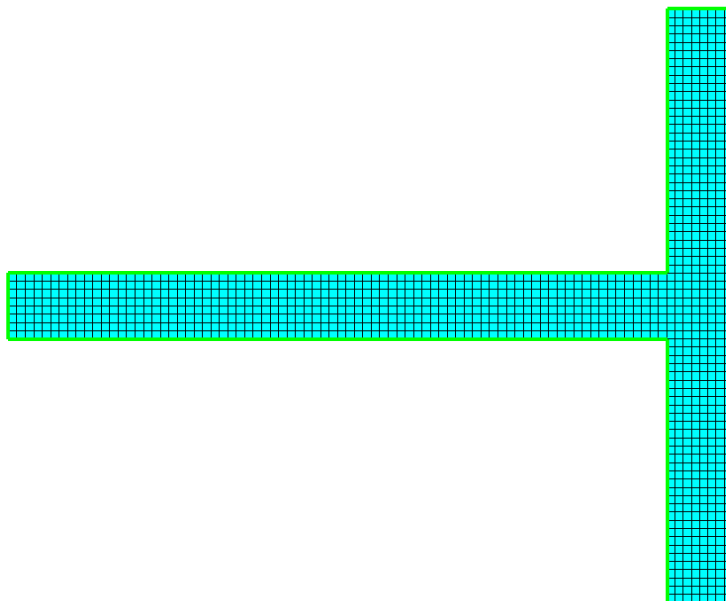


Figure 30.2: Mesh

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation such as file names, time stepping, constants etc.

The simulation is carried out in 2-dimensional Cartesian coordinates. 2nd order bdf time stepping method is selected with 120 steps and with step size of $1e-5$ seconds. Coordinate scaling reduces the size of the mesh and produces a T-shaped micro channel.

Model

Setup

```
Simulation Type = Transient
Steady state max. iter = 20
Timestepping method = BDF
BDF order = 2
Time Step Intervals = 120
Timestep sized = 1e-5
Output intervals = 2
Coordinate Scaling = 1e-5 1e-5 1e-5
```

Apply

In the equation section we choose the relevant equations and parameters related to their solution. In this case we'll have one set of equations (named "Equation") which consists of the Stat Electric equation, the Navier-Stokes equation, and the Advection Diffusion equation.

All three solvers are active in this equation set. Further definitions include, first, that the convection of the species is switched on (besides diffusion). Then that for Navier-Stokes equations the convective term may be left out resulting in laminar Stokes flow, and finally that the electric field is computed by the electrostatics rather than given by the user.

Note that the entry under Electrostatics of "Electric Field = Computed" may not exist in early versions of the ElmerGUI menu definition in the electrostatics.xml file. When the Solver is run, an error may be reported, such as "electrokinetics helmholtz smoluchowski: No external electric field defined for Equation 1". If this occurs, either update your version of Elmer, or generate the .sif file manually, then add this statement `Electric Field = Computed` in the Equation block, and run the solver manually.

When defining Equations and Materials it is possible to assign to the bodies immediately, or to use mouse selection to assign them later. In this case we have just one body and therefore its easier to assign the Equation and Material to it directly.

The electrostatics equation is linear and thus no non-linear iterations are needed. The equation is solved using a fast direct method UMFPack.

The system may include non-linear iterations of each equation and steady state iterations to obtain convergence of the coupled system. It is often a good idea to keep the number of non-linear iterations in a coupled case low. Here we select just a maximum of 30 non-linear iterations for the Navier-Stokes equation.

The advection-diffusion equation does not affect either the electrostatic field or the flow, thus it may be solved only after a converged solution for the previous two equations is available. This is achieved with the `After timestep` definition below. The advected quantity is called Concentration. The advection-diffusion solver uses bubble stabilization method to avoid numerical problems associated with convection type equations.

Model

```
Equation
Name = Equation 1
Apply to Bodies = 1
Electrostatics
Active = True, checked
Electric Field = Computed
Edit Solver Settings
General
Before timestep
Solver specific options
Calculate Electric Field = True
Linear system
Method
Direct = Umfpack
Preconditioning = ILU1
Apply
Navier-Stokes
Active = True, checked
Options
Convect = unchecked, False
Edit Solver Settings
Linear system
Preconditioning = ILU1
Nonlinear system
Max. iterations = 30
Newton
After iterations = 10
Apply
Advection Diffusion
Active = True, checked
Concentration Units = Absolute Mass
Convection = Computed
Edit Solver Settings
General
After timestep = True
Stabilize = False
Bubbles = True
Linear system
Preconditioning = ILU2
Nonlinear system
Max. iterations = 1
Apply
Add
OK
```

The Material section includes all the material parameters. They are divided into generic parameters which are direct properties of the material without making any assumptions on the physical model, such as the mass. Other properties assume a physical law, such as conductivities and viscosity.

We use a material similar to water for this example.

```
Model
Material
  Name = Material 1
  Apply to Bodies = 1
  General
    Density = 1e3
  Electrostatics
    Relative Permittivity = 1.0
  Navier-Stokes
    Viscosity = 1e-03
  Advection Diffusion
    Concentration Diffusivity = 1e-10
Add
OK
```

A Body Force represents the right-hand-side of a equation. It is generally not a required field for a body. No body force is included in this example.

Initial conditions should be given to transient cases, and probably are not needed for steady state solutions. No initial condition is used for this example.

Only one boundary condition may be applied to each boundary and therefore all the different physical BCs for a boundary should be grouped together.

Finally the boundary conditions are defined. The first BC is given for the channel walls. Here, tangential velocity (velocity components 1 and 2) is computed by the Helmholtz-Smoluchowski slip velocity condition, which means that the velocity is computed using the computed electric field and the electro-osmotic mobility as inputs.

```
Model
BoundaryCondition
  Name = channel-walls
  General
    Free text input
    EO Mobility = Real 5e-08
  Navier-Stokes
    Velocity 1 = Variable Pressure
    Real Procedure "Electrokinetics" "helmholtz_smoluchowski1"
    Velocity 2 = Variable Pressure
    Real Procedure "Electrokinetics" "helmholtz_smoluchowski2"
Add
OK
```

The next BC is the inlet condition. We give a potential of 100 Volts and define that there is no flow in y -direction. The analyte concentration at the inlet is defined as a function of time using table format. The concentration is 1.0 up until time instant $3 \cdot 10^{-5}$, is zero after $4 \cdot 10^{-5}$ s and decreases linearly between these two time instants.

```
Model
BoundaryCondition
  Name = inlet-el-A
  Electrostatics
    Potential = 100
  Navier-Stokes
    Velocity 2 = 0.0
  Advection Diffusion
    Dirichlet Conditions
      Concentration
        Variable Time
```

```

        Real
        0.0      1.0
        3.0e-5   1.0
        4.0e-5   0.0
        0.5      0.0
    End
    Add
    OK

```

The final two boundary conditions are for the outlets. Different potentials for these are defined as well as a condition for velocity component.

```

Model
  BoundaryCondition
    Name = outlet-el-B
    Electrostatics
      Potential = 30
    Navier-Stokes
      Velocity 1 = 0.0
  Add
  OK

```

```

Model
  BoundaryCondition
    Name = outlet-el-C
    Electrostatics
      Potential = 0
    Navier-Stokes
      Velocity 1 = 0.0
  Add
  OK

```

The conditions may also be assigned to boundaries in the Boundary condition menu, or by clicking on each boundary with the mouse. Here we use the latter approach as that spares us of the need to know the indexes of each boundary.

```

Model
  Set boundary properties
    Choose Inlet -> set boundary condition inlet-el-A
    Choose Top Outlet -> set boundary condition outlet-el-B
    Choose Bottom Outlet -> set boundary condition outlet-el-C
    Choose Walls -> set boundary condition channel-walls
  OK

```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```

Sif
  Generate
  Edit -> look how your command file came out

```

Before we can execute the solver we should save the files in a directory. The ElmerGUI project includes all the files needed to restart the case.

```

File
  Save Project

```

After we have successfully saved the files we may start the solver.

```

Run
  Start solver

```

A convergence view automatically pops up showing relative changes of each iteration.

When there are some results to view we may start the postprocessor also.

```
Run
Start ParaView
```

Results

Due to the number of the time steps the simulation may take around 15 seconds.

You may inspect the results with Paraview or with ElmerVTK.

In Figure 30.3 the obtained concentration distribution is presented.

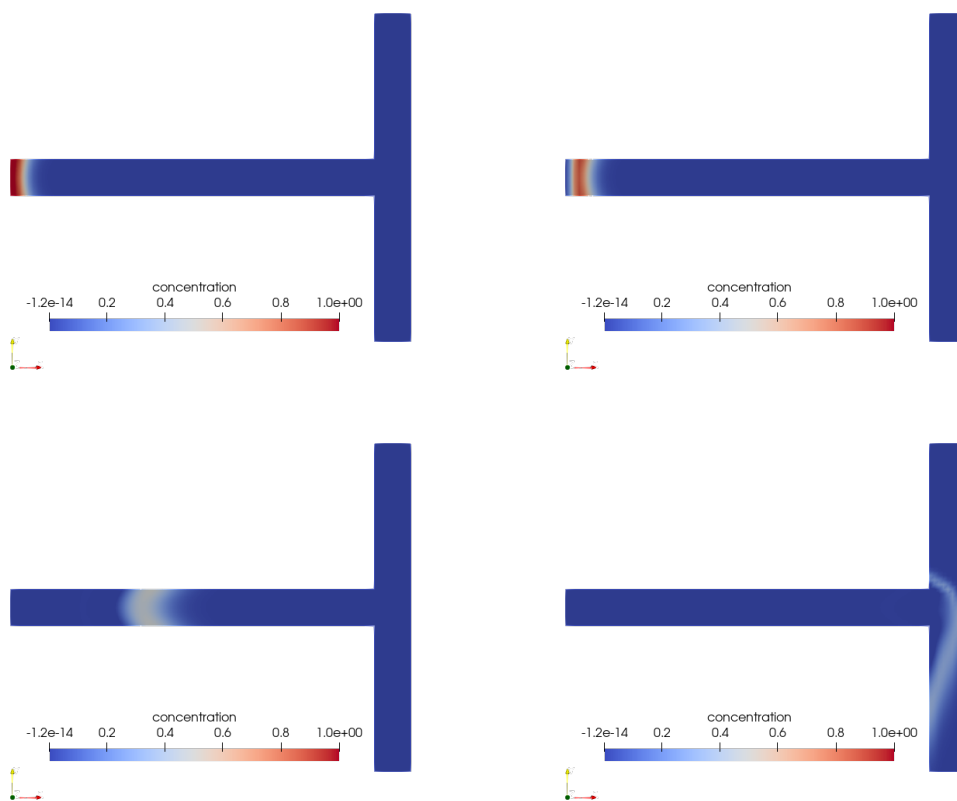


Figure 30.3: Concentration at time: 0, 2, 15, and 45 ticks

Tutorial 31

Asymmetrical Conductor with a Hole TEAM Workshop Problem 7

Directory: TEAM7

Solvers: CoilSolver, WhitneyAVSolver, MagnetoDynamicsCalcFields

Tools: ElmerGUI

Dimensions: 3D, Transient

Author: Jonathan Velasco

Case definition

The Testing Electromagnetic Analysis Methods (TEAM) Problem 7 is a validation case consisting of a thick aluminium plate with a hole, which is placed eccentrically in a non-uniform magnetic field. The field is produced by a sinusoidal current flowing through a coil that hovers over the plate as shown in Fig. 31.1. Due to asymmetry a simplification of the model becomes unavailable and a full 3D model is needed. The goal of the model is to demonstrate the basic setup of a 3D eddy current problem.

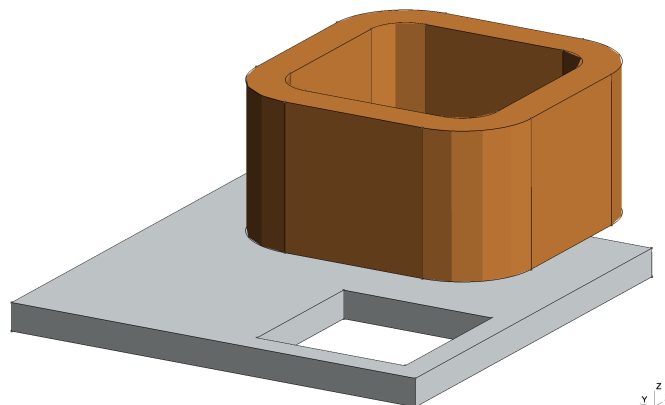


Figure 31.1: TEAM Problem 7 Geometry Description

The full description of the given problem can be found in the COMPUMAG website under:

TEAM Problem 7

Mesh and tailored .sif file can be found in our Github repository under the TEAM7 directory:

Elmer Electromagnetics Tutorials

Meshing: From Gmsh to ElmerGrid

The TEAM7 Mesh can be found in our Github repository under the TEAM7 directory:
Elmer Electromagnetics Tutorials

However, you may also want to import your own mesh from another software. Under the same GitHub repository you'll find the .geo file created in Gmsh (gmsh/TEAM7.geo). Note, this mesh is generated with Gmsh but ElmerGrid parses several formats into ElmerSolver format. (e.g., .ansys, .inp, .fil, .mphtxt ...).

To mesh .geo file please run the following cmd/terminal command

```
gmsh TEAM7.geo -3
```

where the `-3` implies 3D meshing. This will save the mesh as `TEAM7.msh`
If further mesh refinement is needed:

```
gmsh TEAM7.msh -refine
```

The command performs uniform mesh refinement. You can also perform barycentric refinement with command: `-barycentric_refine`.

Once the mesh is saved you can parse it into ElmerSolver by using the following command

```
ElmerGrid 14 2 TEAM7.msh
```

where 14 denotes the input mesh format (Gmsh) and 2 the output mesh format (ElmerSolver).
The mesh is ready to be used by ElmerSolver either using command line or ElmerGUI.

Necessary Solver Modules

The TEAM7 magnetodynamics problem can be solved using two modules: the *Coil Current Solver* and the *WhitneyAV-Solver*. The field solution can be retrieved using the *MagnetoDynamicsCalcFields* module.

The *Coil Current Solver* solves for the irrotational electric field associated with the gradient of the electric scalar potential. Hence, the current density is calculated and may be used as an external source (Body Force) when solving the magnetodynamics problem. The magnetodynamics problem is then solved throughout the whole domain (coil, plate and air) using the *WhitneyAVSolver* and the field solution such as the magnetic flux density, magnetic field strength among other variables can be obtained for postprocessing visualization using the *MagnetoDynamicsCalcFields* module.

ElmerGUI Project

Start ElmerGUI from command line or by clicking the icon in your desktop. Here we describe the essential steps in the ElmerGUI by writing out the click-by-click procedure. Indentation generally means that the selections are done within the window chosen at the higher level.

Before setting up the model, let us configure ElmerGUI's menu and necessary settings related to this particular model by using the top ribbon in the GUI (Fig. 31.2)



Figure 31.2: Top Menu

For simplicity the project directory is presented under ElmerGUI_TEAM7. The model is solved using modules *Coil Current Solver*, *WhitneyAVSolver* and *MagnetoDynamicsCalcFields*, which are found under the *coilsolver.xml* and *magnetodynamics.xml* menus. As of the current version of the code, these menus are not added automatically and need to be added manually. Let us setup the project directory, load the mesh (TEAM7) in ElmerSolver format (.mesh.*) and load the necessary menus. See Fig

```
File
  New project ...
  Project directory
```

```
Select project dir -> ElmerGUI_TEAM7
Geometry input
Elmer mesh -> TEAM7
Extra EDFs to be added
Add -> coilsolver.xml
Add -> magnetodynamics.xml
OK
```

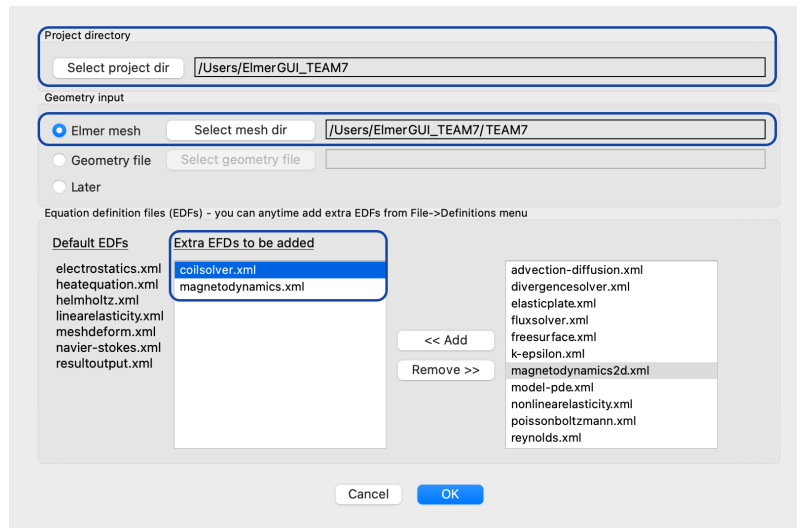


Figure 31.3: New Project Window

Adding additional menus can be avoided by permanently moving the .xml files from $\$ELMER_HOME/install/share/ElmerGUI/edf$ to $\$ELMER_HOME/install/share/ElmerGUI/edf$

Model Preferences

The TEAM Problem 7 is solved by imposing a transient sinusoidal current that generates a magnetic field. Due to the time-dependency nature of the problem let us setup the model time preferences by clicking on ElmerGUI on the top ribbon

```
ElmerGUI
Preferences
Simulation Type = Transient
Timestep interval = 16
Timestep sizes = 0.0025
Close
```

The timestep interval and size used are meant to accommodate two periods at a frequency of 50Hz.

Header

Check keywords warn

MeshDB .

Include path

Results directory

Free text

Simulation

Max.output level Steady state max.iter

Coordinate system Timestepping method

Coordinate mapping BDF order

Simulation type Timestep intervals

Output intervals Timestep sizes

Coordinate Scaling Angular Frequency

Solver input file Post file

Free text

Constants

Gravity Boltzmann

Stefan Boltzmann Unit charge

Figure 31.4: Preferences Setup Window

Solution procedure

At this point the mesh (Fig. 31.5) has been loaded, the model setup has been configured and the necessary menus have been added to the GUI. We may now proceed to set up the physics of the TEAM Problem 7.

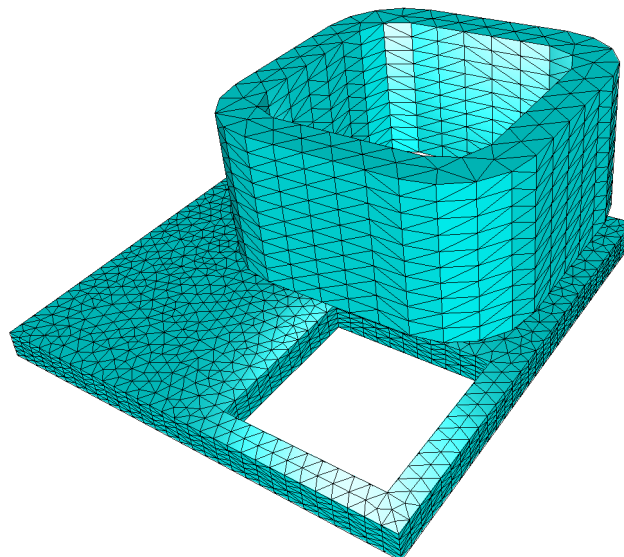


Figure 31.5: TEAM 7 Mesh.

Let us configure the equations. Two sets of equations are needed in this problem. The first set of equations will apply to the Coil domain alone whereas the second set applies to the Plate and Air domain. The reasoning behind it lies in the

fact that we will solve the current direction before the simulation begins, using it as a source in the magnetodynamics problem.

The coil equation are setup in the following manner

Model

```
Equation -> Add
  Name = Equation Coil
  Apply to Bodies = 1
```

```
CoilSolver
  Active = on
  Edit Solver Settings
    Solver specific options
      Coil Closed=on
      Desired Coil Current = -2742
      Normalize Coil Current=on
      Coil Normal (3-vector) = 0 0 1
      Narrow Interface=on
      Fix Input Current Density=on
    General
      Execute solver -> Before Simulation
  Apply
```

```
MgDyn
  Active = on
  Edit Solver Settings
    Solver specific options
      Use Elemental CoilCurrent=on
  Apply
```

```
MgDynPost
  Active=on
  Edit Solver Settings
    Solver specific options
      Specify fields to compute
        Calculate Current Density=on
      What kind of fields to compute
        Discontinuous bodies=on
    General
      Execute solver -> Before Saving
  Apply
```

```
Add
OK
```

The configuration of the menus will appear also in any new equation added. For this reason, we only need activate the solvers needed and apply them to the respective bodies

```
Model
Equation -> Add
  Name = Equation Plate and Air
  Apply to Bodies = 1 2

  MgDyn
    Active = on

  MgDynPost
    Active=on

Add
OK
```

Elmer contains an extensive material library however, for this exercise we will define some user defined material properties. In this example we only need two materials: Air and Aluminium which are applied on. Due to the fact that the current is known a priori, the current density will be used as an external current source enabling the coil to be part of the non-conducting domain (Air).

```
Model
Material -> Add
  Name = Air
  Apply to Bodies = 1 3
  MgDyn
    Relative Permeability=1
Update
OK
```

```
Model
Material -> Add
  Name = Aluminium
  Apply to Bodies = 2
  MgDyn
    Relative Permeability=1
    Electric Conductivity =3.526e7
Update
OK
```

A Body Force represents the right-hand-side of a equation. In this case we want to re-use the computed current density (by the Coil Solver) and use it in our MagnetoDynamics problem. By activating the *Use Elemental CoilCurrent=on* checkbox in the MgDyn solver menu, this action is performed. However, since the current is sinusoidal, we will set up a feature that is found within the Body Force menu.

```
Model
Body Force
  Name = External CoilCurrent Source
  Apply to Bodies = 1
  Current Density Multiplier= Variable "time"; Real LUA "cos(2*math.pi*50*tx[0])"
Add
OK
```

Note that the frequency is 50Hz. Hence $\cos(2 \pi f t)$, where f is the excitation frequency.

Lastly, at infinity the magnetic vector potential is set to zero

```
Model
  BoundaryCondition -> Add
    Name = Infinity
    Apply to Boundaries = 6
    MgDyn
      AV {e} = 0.0

  Add
  Ok
```

For the execution ElmerSolver needs the mesh files and the command file. We have now basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

```
Sif
  Generate
  Edit -> Double check file has been generated accordingly to specs
```

Before we can execute the solver save the the project.

```
File
  Save Project
```

After we have successfully saved the files we may start the solver.

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration.
When there are some results to view we may start the postprocessor also.

```
Run
  Start ParaView
```

Results

Note that this model's purpose is two fold. First, our goal is to setup a basic 3D eddy current problem. Second, we use this model for validation purposes.

You may inspect the results with Paraview or with ElmerVTK.

The following image have been created using Paraview.

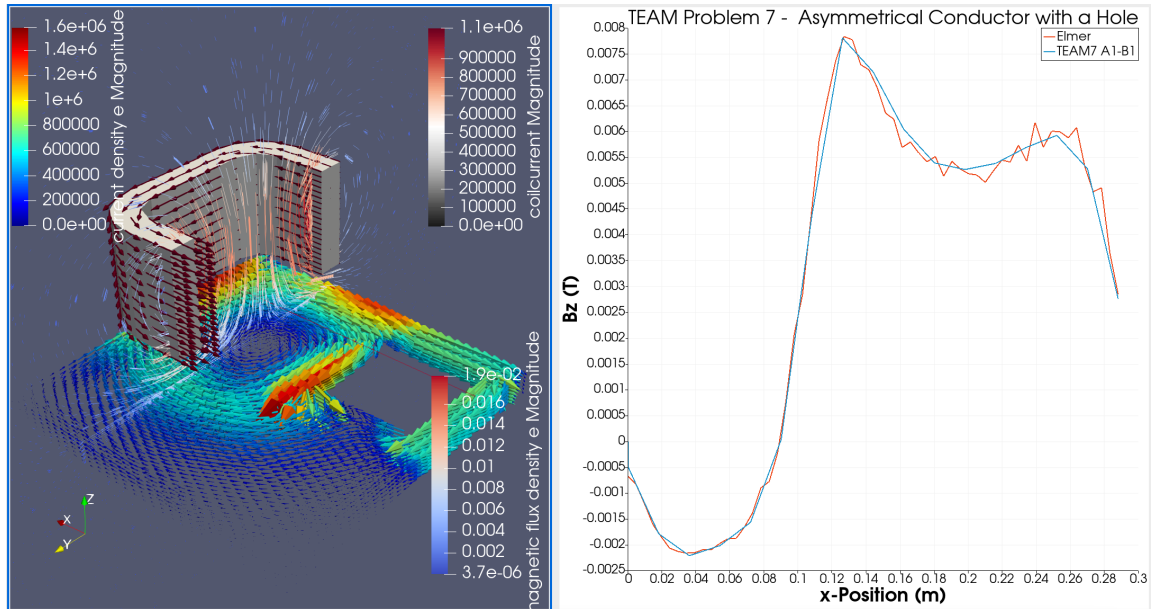


Figure 31.6: Postprocessing: Color map including induced currents on plate (Left), Line Plot to validate Elmer's results against TEAM7's

Index

- advection-diffusion.xml, 187
- AdvectionDiffusion, 186

- Boussinesq, 165

- capacitance, 74, 85
- clipping with ElmerVTK, 24
- clipping with Paraview, 27
- CoilSolver, 193
- Convection Coefficient, 181
- Convection Velocity 1, 181
- Convection Velocity 2, 181
- Convection Velocity 3, 181

- Diffusion Coefficient, 181
- Discontinuous Bodies, 94

- ElasticSolve, 59, 158
- electro-osmotic mobility, 190
- Electrokinetics, 186
- Electrokinetics solver, 187
- electrostatics.xml, 187
- ElmerGrid, 90, 122
- ElmerGUI, 7, 29, 34, 40, 45, 51, 55, 59, 64, 69, 74, 79, 85, 90, 97, 102, 107, 112, 118, 122, 136, 141, 148, 152, 158, 165, 170, 176, 181, 186, 193
- ElmerVTK, 20
- External Field, 182

- Field, 182
- Field Flux, 182
- Field Source, 182
- Flow Solver, 187
- FlowSolve, 118, 122, 136, 141, 148, 152, 158, 165, 186
- FlowSolver, 142, 176

- Gmsh, 97, 102

- Heat Solver, 9, 46
- heatequation.xml, 9, 46, 142
- HeatSolve, 29, 34, 40, 45, 141, 152, 165
- HeatSolver, 7, 142, 170, 176
- Helmholtz, 107
- Helmholtz Solver, 108
- Helmholtz-Smoluchowski, 186
- helmholtz.xml, 108

- KESolver, 136

- magnetodynamics.xml, 90
- MagnetoDynamics2D, 97

- MagnetoDynamics2DHarmonic, 102
- MagnetoDynamicsCalcFields, 193
- MATC, 149
- MeshSolve, 158
- MgDyn, 92
- MgDynPost, 92
- MgHarm, 92
- ModelPDE, 181

- Navier-Stokes Solver, 124
- navier-stokes.xml, 124, 142, 187
- netgen, 7, 79
- nglib, 170, 176
- nonlinearelasticity.xml, 59

- OpenCascade, 7

- Paraview, 25
- passive elements, 45
- Polyline Coordinates, 126
- Polyline Divisions, 126

- Rayleigh-Benard, 165
- Reaction Coefficient, 181
- Result Output Solver, 113
- resultoutput.xml, 113
- Reynolds Number, 148
- Robin Coefficient, 182

- save a picture with ElmerVTK, 23
- save a picture with Paraview, 26
- SaveLine Solver, 124
- saveline.xml, 124
- SmitcSolver, 64, 69
- Smoluchowski, 190
- Stat Elec Solver, 187
- StatElecSolve, 186
- StatElecSolver, 74, 79, 85
- StressSolve, 51, 55

- Time Derivative Coefficient, 181

- VectorHelmholtz, 112
- vectorhelmholtz.xml, 113
- von Karman, 148
- vortex shedding, 148

- WhitneyAVHarmonicSolver, 90
- WhitneyAVSolver, 193