Elmer GUI Tutorials

CSC – IT Center for Science

August 3, 2020

Elmer GUI 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 field 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 used by advanced users.

The present manual corresponds to Elmer software version 8.4. Latest documentations and program versions of Elmer are available (or links are provided) at `http://www.csc.fi/elmer`.

Copyright information

Contents

Instructions for the GUI tutorials

Here are some instructions for following the GUI tutorials:

- All the needed input files should be available among the `ElmerGUI/samples` directory that should have come with the installation. Look under a subdirectory named after the suffix of the sample file.
- The instructions written in `verbatim` refer to operations with the GUI. Intendation means step in the menu hierarchy. The instructions should not be mixed with those in the 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 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 structures exist or not.
- After having once defined the case you may go to the working directory and launch ElmerSolver from command-line. There you may edit the .sif file to alter the parameters.
- Manual alteration to the sif file will not be communicated to the ElmerGUI project. All editions will be overrun by the GUI when saving the project.
- It is assumed that the default method for visualization is Paraview (or ViSit) using `vtu` format. `ElmerPost` is no longer available in all installations and the VTK widget that comes with ElmerGUI is instable and have also been eliminated from many installations. If you have access to these tools and still insist using either of them change the default suffix for output from `.vtu` to `.ep`. Unfortunately, the tutorial does not still use Paraview for all the cases.
- The cases have been run a number of times but errors are still possible. Reporting them to elmeradm@csc.fi, for example, is greatly appreciated.

Tutorial 1

Heat equation – Temperature field of a solid object

Directory: TemperatureGenericGUI
Solvers: HeatSolve
Tools: ElmerGUI,netgen,OpenCascade
Dimensions: 3D, Steady-state

Problem description

This tutorial tried 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. At some part of the boundary the temperature is fixed. Mathemetically the problem is described by the Poisson equation

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

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 the 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.

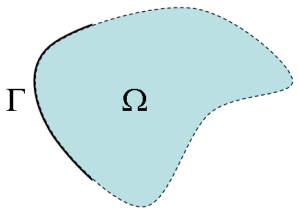

Figure 1.1: Generic object being heated

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 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 you lack 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 a 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 in the number of element in the `Model summary`. With netgen the default setting generates 8371 nodes and 36820 tetrahedral elements. Visual inspection rewiels that the mesh is not quite satisfactory in geometric accuracy. We choose to modify the mesh by altering the settings in the following way.

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

The meshing a take a minute or two. The modified mesh should include 16159 nodes and 65689 tetrahderal elements and be more appieling to the eye. In order to affect the mesh density study the command-line options of the netgen manual. Here we continue with the default mesh.

We want to set the temperature at the inside of the holes and in that aim you may join the three boundaries (see figure 1.2). For that aim we may choose the six pieces that constitute the boundaries as shown in the picture by pressing the `Ctrl`-key down.

```
Mesh
  Unify Surface
```

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 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 the to 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, whereas the active boundary is chosen graphically.

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

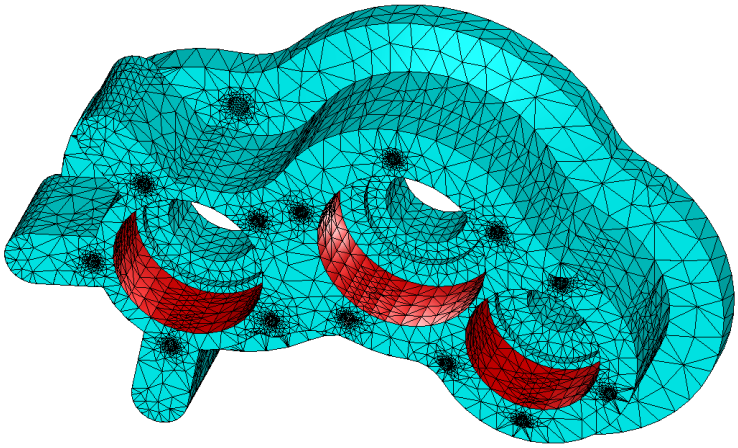

Figure 1.2: The computational mesh showing the three joined boundaries

```
Model
  Equation
    Add
      Name = Heat Equation
      Apply to bodies = Body 1
      Heat Equation
        Active = on
      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 heat conductivity. We choose Aluminium from the Material library which automatically sets for the needed material properties.

```
Model
  Material
    Add
      Material library
        Aluminium
      Apply to bodies = Body 1
      Add
      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 = 0.01
      Apply to bodies = Body 1
      Add
      OK
```

No initial conditions are required in steady state case.

In this case we have only one boundary and set it to room temperature. First we create the boundary condition

```
Model
  BoundaryCondition
    Add
      Heat Equation
        Temperature = 293.0
      Name = RoomTemp
      Add
      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 this boundary.

```
Boundary condition
  RoomTemp
```

For the execution ElmerSolver needs the mesh files and the command file. We have know 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 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 setting, always remember to regenerate the sif file and save the project before execution.

Postprocessing

To view the results we use Paraview for the visualization,

```
Run
  Paraview
```

The default configuration shows just the object. To color the surface with the temperature choose the correct field. The maximum temperature should be about 586.5 K. You may turn on opasity in order to see through the object, 10-20% is a good value. This way you'll able to see some isosurfaces that you might want to define. Some examples of the visualizations may be seen in figure 1.3. Note that the pictures here were generated by the obsolite VTKPost tool within ElmerGUI and therefore does not look like the results in Paraview.

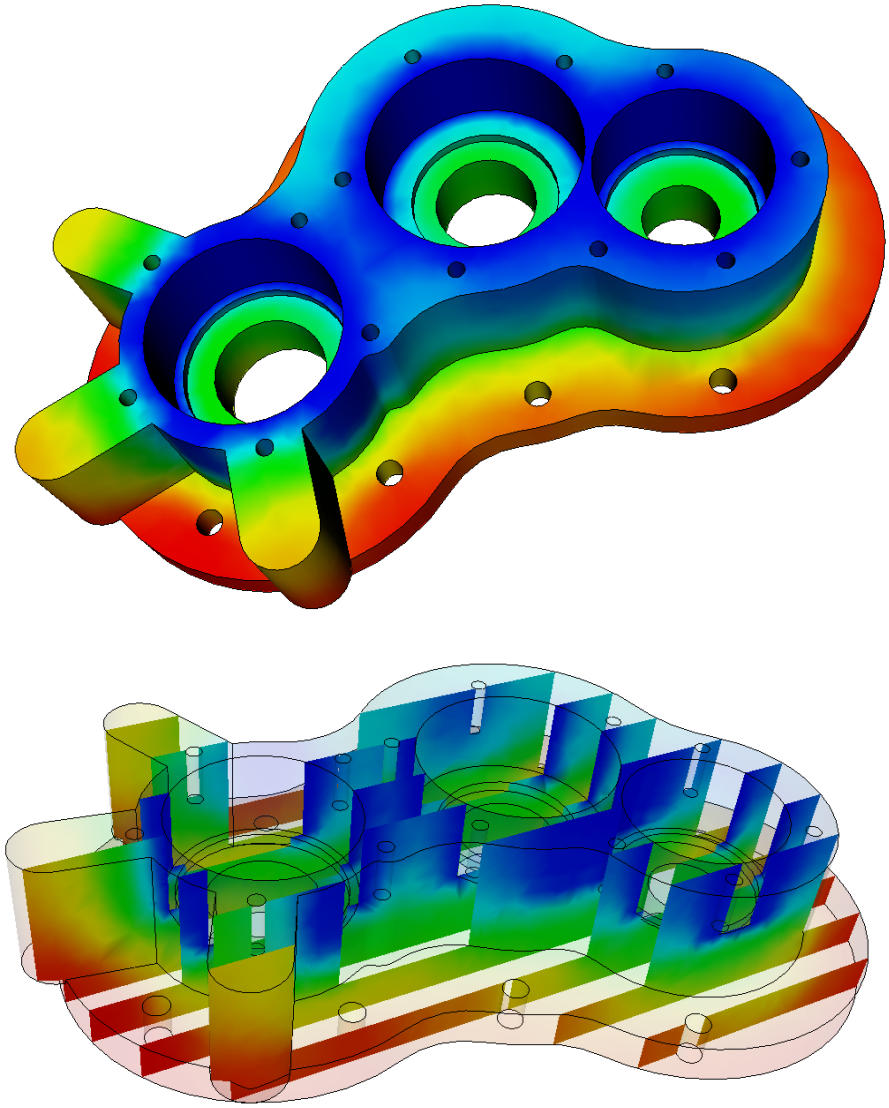

Figure 1.3: The temperature distribution of the solid object domain.

Tutorial 2

Generic scalar PDE on 3D angle domain

Directory: ModelPDE3D
Solvers: ModelPDE
Tools: ElmerGUI
Dimensions: 3D, Steady-state

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 some own code development.

The problem is a simple 3d structure `winkel.grd` that can be charectized 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

`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 Coeffficient` `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. 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 `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 2.1. 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 `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 Model PDE.

When defining Equations and Materials it is possible to assign the to 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.

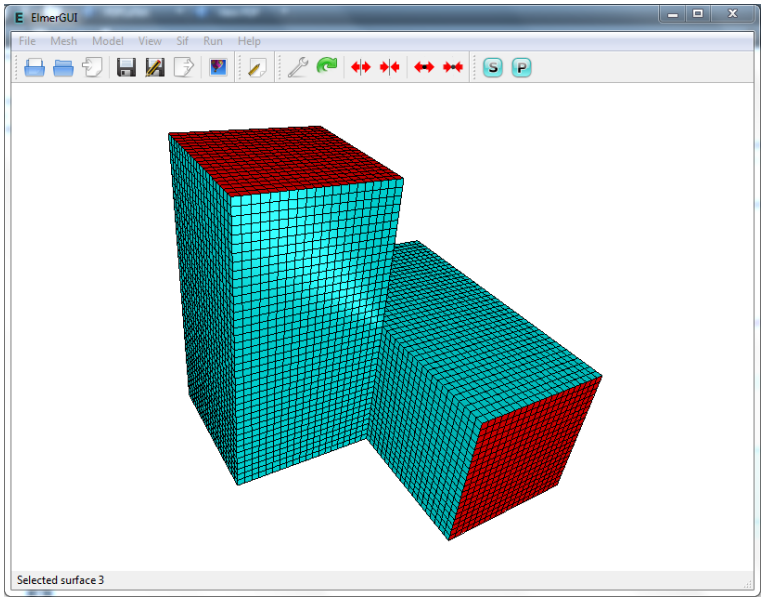

Figure 2.1: The finite element mesh in ElmerGUI

```
Model
  Equation
    Add
      Name = Model PDE
      Apply to bodies = 1
      Model PDE
        Active = on
  Apply
  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
    Add
      Name = Ideal
      Apply to bodies = 1
      Model PDE
        Diffusion Coefficient = 1.0
    Apply
    OK
```

A Body Force represents the right-hand-side of a equation,

```
Model
  Body Force
    Add
      Name = Source
      Field Source = 1.0
      Apply to bodies = 1
    Apply
    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,

```
Model
  BoundaryCondition
    Add
      Model PDE
        Field = 0.0
      Name = Zero
    Apply
    OK
```

For the execution ElmerSolver needs the mesh files and the command file. We have know 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 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 equatios 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 becore eventually oscillatory. To eliminate that some bubbles or p-elements may be needed. You may play around with element settings, for exammple use `Element = p:2` or `Element = n:1 b:1` etc.

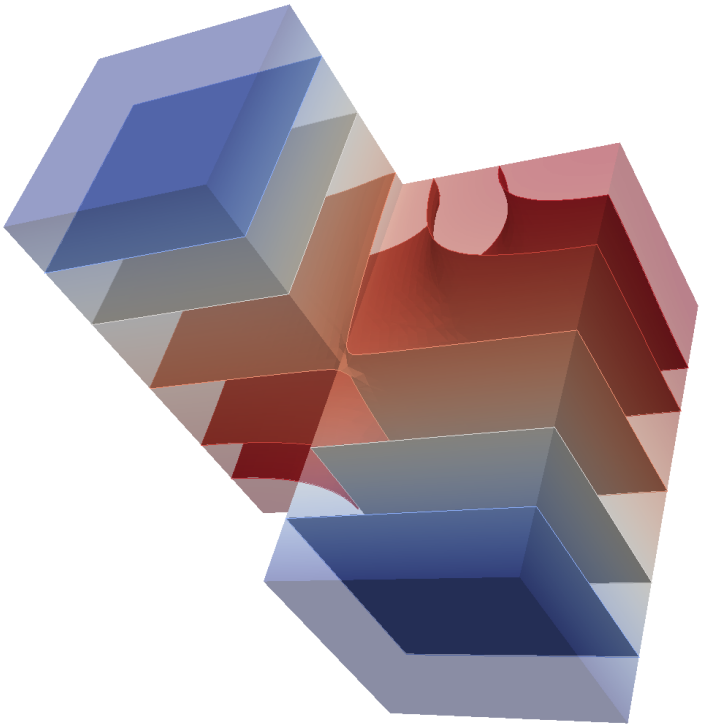

Figure 2.2: The field values of the structure as visualized with Paraview. Isosurfaces are defined for field values 0.5, 1.0, 1.5, 1.8 and 1.9, respectively.

- 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 nonlinearities that might not be well handled by the fixed point iteration scheme. For more demanding nonlinearities Newton linearization or other technniques 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 3

Linear elasticity equation – Loaded elastic beam

Directory: ElasticBeam3D
Solvers: StressSolve
Tools: ElmerGUI
Dimensions: 3D, Steady-state

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 $10 \cdot 10^9 \mathrm{N/m}^2$, and density 550 kg/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 phenomenom known as locking.

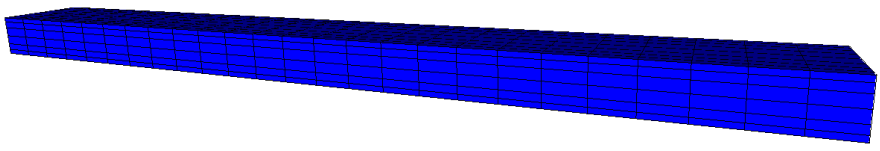

Figure 3.1: The mesh used in the computations

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 nonlinear 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 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 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 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 know 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 meaningfull output since the 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 3.2

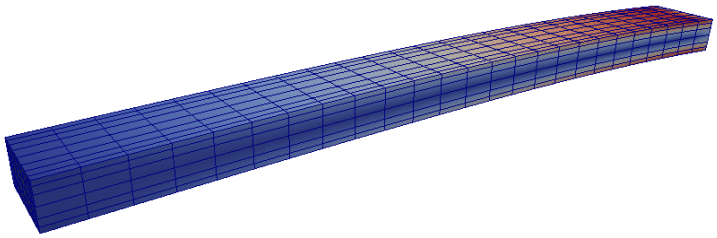

Figure 3.2: The displaced shape of the elastic beam colored with the von Mises stresses.

Note that the displacement 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 nonlinearities – 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 negitive 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 4

Nonlinear elasticity equation – Loaded elastic curve

Files: u_turn.grd
Solvers: ElasticSolve
Tools: ElmerGUI
Dimensions: 3D, Transient

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 solver capable of dealing with large displacement.

Solution procedure

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 of no importance so we could as well scan through a set of steady state loading conditions. We choose the timestep 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.

```
Model
  Setup
    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 we are happy to use the defaults. One may however, try out different preconditioners (ILU1,...) or more efficient iterative methods (BiCGStabl), for example.

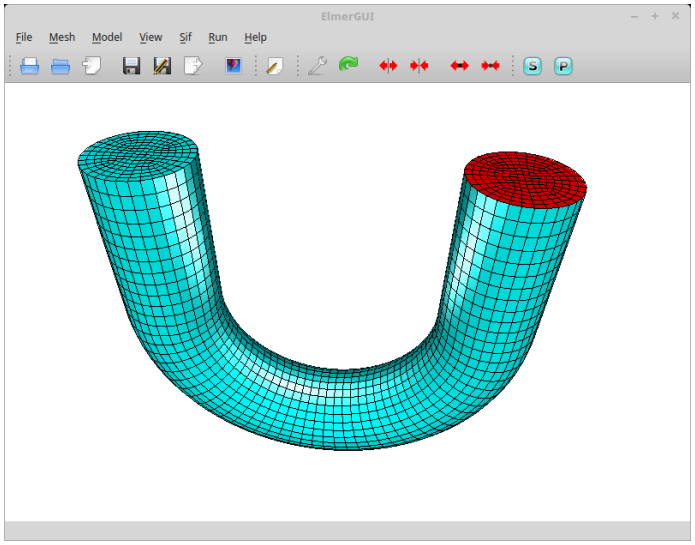

Figure 4.1: The mesh used in the computations as shown in ElmerGUI.

```
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 trough the material parameters of the various solvers to ensure that the properties are indeed as they should be. Any consistant 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 displacent components are set to zero. To type in the multiline expressions for the boundary condition just press `Enter` `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 know 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 timesteps the convergence will be shown for each timestep. You can start looking at the results already after a few timesteps. 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 make the equation nonlinear.

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.

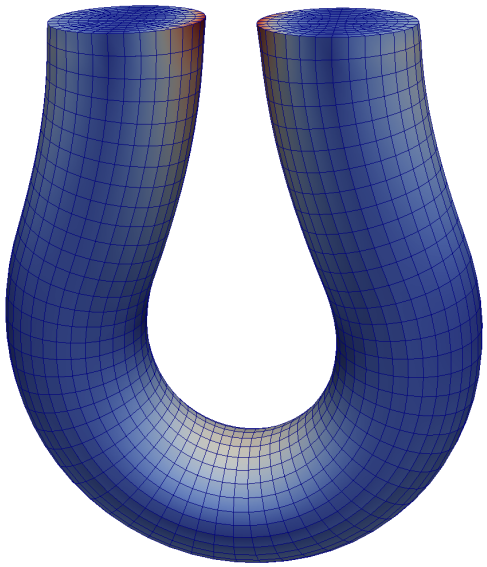

Figure 4.2: The final state of the deformed curve. Color shows the resulting von Mises stresses.

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

$$\begin{aligned} u_x = &\quad (\cos(\phi) - 1)x - \sin(\phi)y) \\ u_y = &\quad (\cos(\phi) - 1)y + \sin(\phi)x) \end{aligned} \tag{4.1}$$

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 revolation of 180 degrees is performed in 1 second.

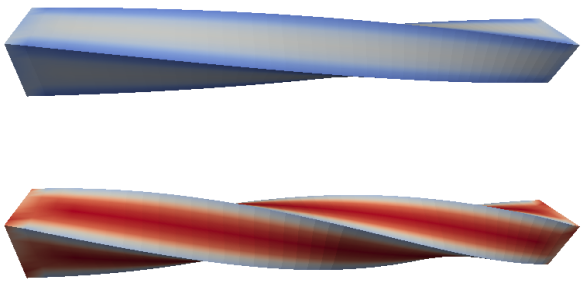

Figure 4.3: A square profile being rotated from one end by 90 and 180 degrees. Color shows the von Mises stresses.

Tutorial 5

Smitc solver – Eigenmodes of an elastic plate

Directory: ElasticPlateEigenmodesGUI
Solvers: SmitcSolver
Tools: ElmerGUI
Dimensions: 2D, Eigenmode

Problem description

For thin elastic structures it is often advicable 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 shoud make sure that the menus for Smitc solver are present. If not, they may be found in file

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

To load these definitions do the following

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

To see what kind of new menu structures you got 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 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.

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

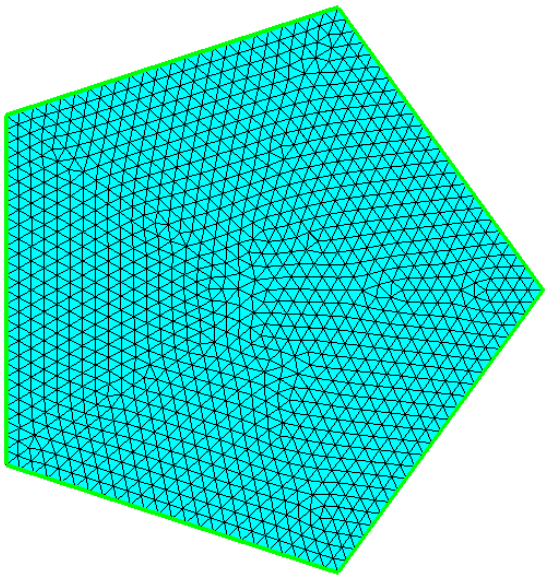

Figure 5.1: The finite element mesh in ElmerGUI

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 5.1.

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 the to 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 performes 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 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 Youngs 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 derivaties 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 5.1. Note that some eigenmodes are degenerated but as the finite element mesh is not perfectly symmetric there will be minor differencies in the eigenvalues.

Table 5.1: 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 choosed for vtu ouput the mode `Eigen Analysis = True` then each file includes one eigenmode. Then the eigenmodes may be treated as timesteps. In figure 5.2 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.

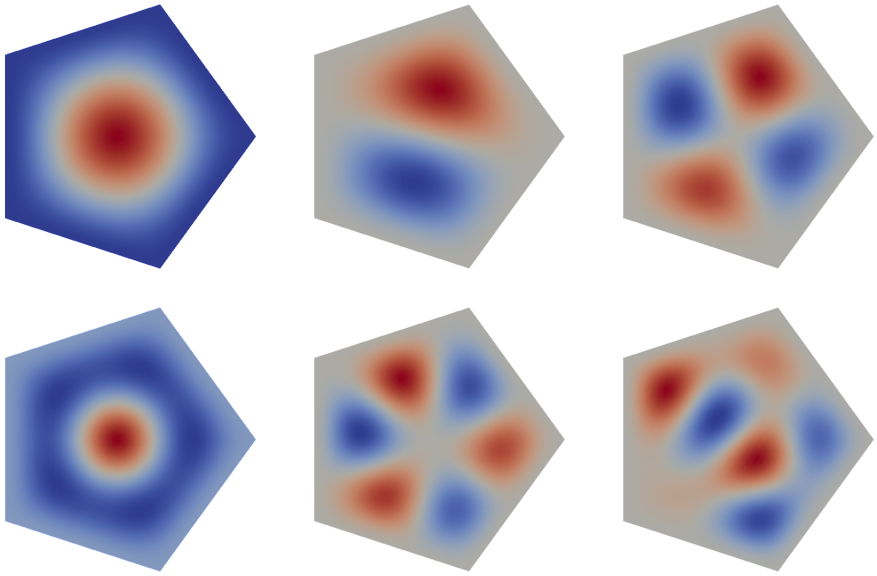

Figure 5.2: The 1st, 2nd, 4th, 6th, 7th and 9th eigenmode of the plate. The displacement in the normal direction (first component) is shown.

Tutorial 6

Electrostatic equation – Capacitance of two balls

Directory: CapacitanceOfTwoBalls
Solvers: StatElecSolver
Tools: netgen,ElmerGUI
Dimensions: 3D, Steady-state

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 [**?**, Ch. A.3]:

$$C_{12} = 4\pi\varepsilon \frac{a^2}{d}\left(1 + \frac{a^2}{d^2 - 2a^2} + \frac{a^4}{d^4 - 4d^2a^2 + 3a^4} + \dots\right) \tag{6.1}$$

and the self-capacitance from

$$C_{10} = C_{20} = 4\pi\varepsilon a\left(1 - \frac{a}{d} + \frac{a^2}{d^2 - a^2} - \frac{a^3}{d^3 - 2da^2} + \dots\right) \tag{6.2}$$

Let's mark $\tilde{C} = C/\varepsilon$. 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

In this case 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. Ultimaterly 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 `Refinemenent`) 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 opended by ElmerGUI.

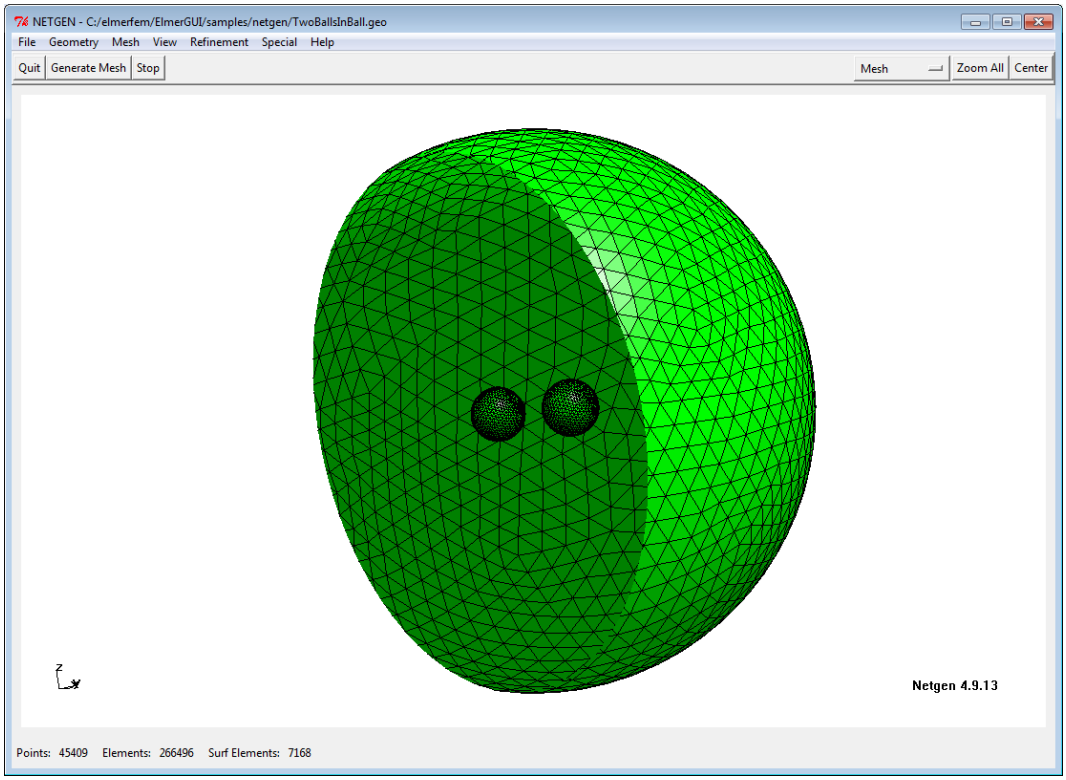

Figure 6.1: Surface mesh for the two inner balls as seen in Netgen

The order of the mesh using nodal elements may be increased by `ElmerGrid`. Assuming the mesh would recide 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 recide 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
```

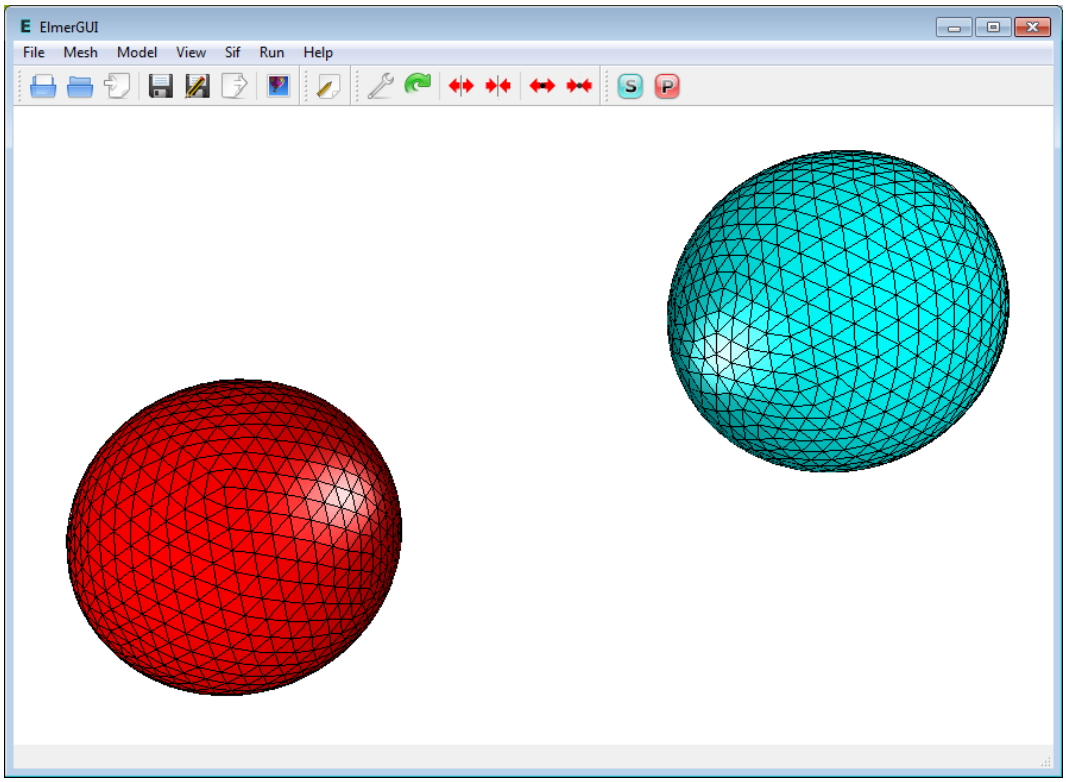

Figure 6.2: The mesh with one highlighted ball as seen in ElmerGUI

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. 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 the to 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
    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 know 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 nonlinear level was really obtained. The norm of the solution should be?

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

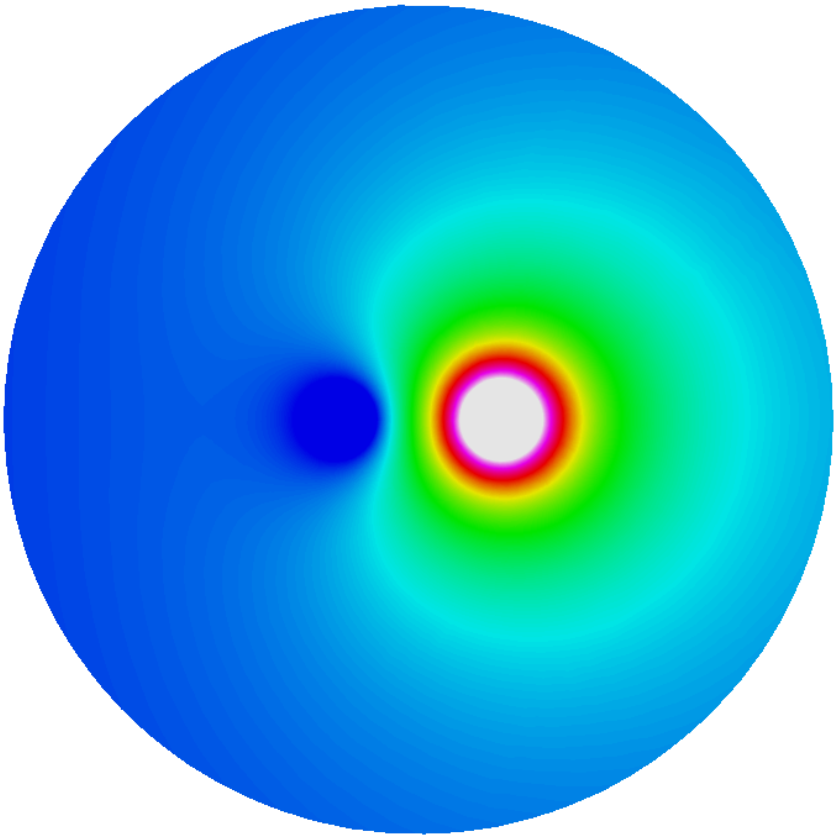

Figure 6.3: The electrostatic potential on the clipping plane. This is the latter of the two symmetric configurations where the unit voltage is applied to one ball and zero voltage to the other, respectively.

Tutorial 7

Magnetic field induced by harmonic current in a wire

Files: wire.grd
Solvers: WhitneyAVHarmonicSolver
Tools: ElmerGrid, ElmerGUI
Dimensions: 3D, Harmonic

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 $R = 0.01$ 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.

$$\vec{A} = \begin{cases} -\frac{I\mu_0}{2\pi}\ln(r/R)\vec{e}_z, & r > R \\ -\frac{I\mu_0}{4\pi R^2}(r^2 - R^2)\vec{e}_z, & r \leq R \end{cases} \tag{7.1}$$

resulting to a magnetic flux density

$$\vec{B} = \begin{cases} \frac{I\mu_0}{2\pi r}\vec{e}_\phi, & r > R \\ \frac{I\mu_0}{2\pi R}\frac{r}{R}\vec{e}_\phi, & r \leq R \end{cases} \tag{7.2}$$

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

$$|\vec{B}| = \frac{1}{2}\mu_0 R\sigma|\vec{E}| \tag{7.3}$$

Using elecric conductivity of copper (σ=59.59e6 A/m^{2}V) we obtain 0.0374 T.

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

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 -> magnetodynamics.xml
```

The additional definitions should recide 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 `samples` directory were it recides.

```
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.

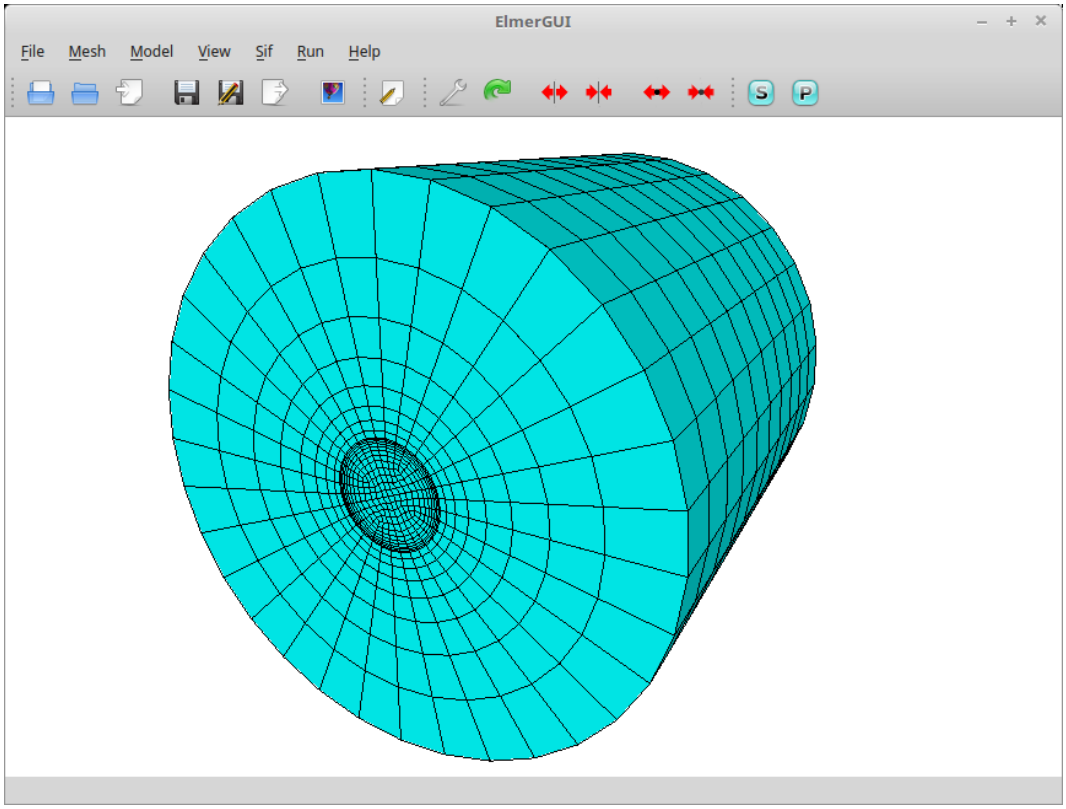

Figure 7.1: The mesh for the wire and surrounding air as seen in ElmerGUI

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 1000 times too large. Hence we apply scaling in this menu. Also we want to use an harmonic equation that requires the frequency to be set somewhere.

```
SetUp
  Coordinate Scaling
```

```
    1.0e-3
  Angular Frequency
    1.0e5
```

Currently the permeability of vacuum is not given in the ElmerGUI. To set other than the default value for it (in case using a different unit system), 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 `MgDynHarm` 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 the to 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 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 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 = MgDynHarm
      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 Strenth = 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.

```
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
    MgDynHarm
      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 as the length of the wire is 0.01 m.

```
Model
  BoundaryCondition
    Name = Voltage
    MgDynHarm
      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 extenal BCs we set the boundary non-axial components of the vector potential to zero.

```
Model
  BoundaryCondition
    Name = AxialField
    MgDynHarm
      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 nonlinear 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 were present some results of the computations. The visulalization is done using Paraview and the `vtu` format. For optimal visualization use the `Discontinuous Bodies` flag turned on both on the calculation of the fields and while saving the results in `vtu` format. These flags ensure that the solution is enforces continuous over 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 7.2. The imaginary part dominates in the amplitude the (0.524 T vs. 0.107 T). The corresponding vector field of the magnetic flux density is depicted in Figure 7.3.

Why are the results so far of 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. The maximum magnetic flux density from the computations is 0.0367 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 anymore and the results become questionable.

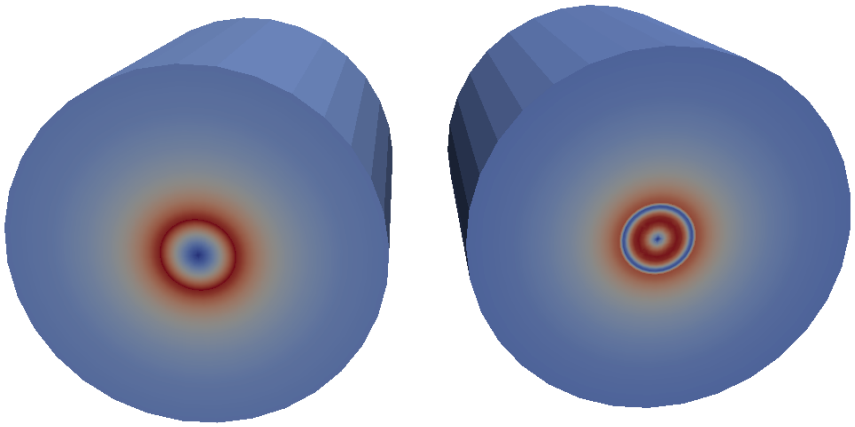

Figure 7.2: Imaginary and real parts of the magnetic field strength.

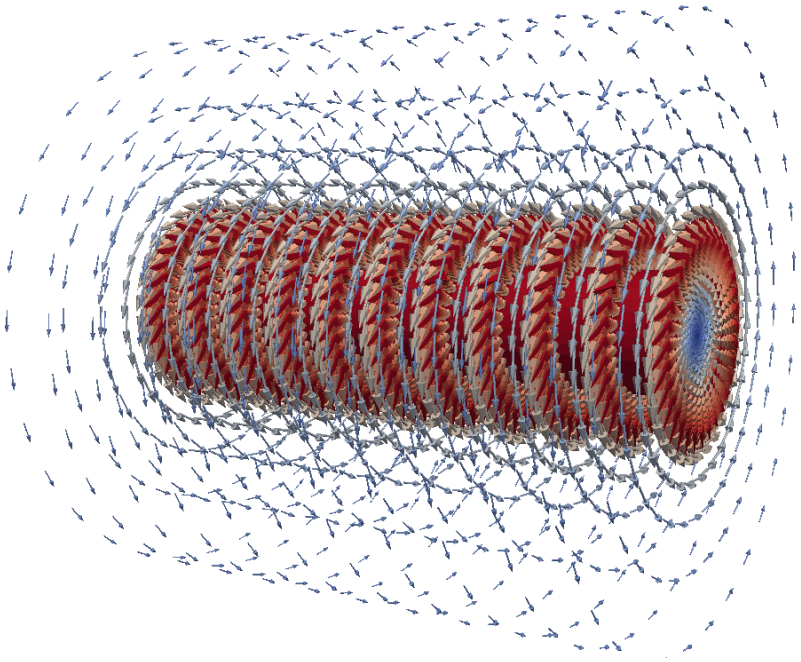

Figure 7.3: Imaginary part of the magnetic field strength plotted with vectors scaled by the field values.

Tutorial 8

Magnetostatics – Magnetic field resulting from a permanent magnet

Directory: Horseshoe
Solvers: MagnetoDynamics2D
Tools: Gmsh, ElmerGUI
Dimensions: 2D, Steady-state

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 farfield 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 recide 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 recides.

```
File
  Open -> horseshoe.msh
```

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

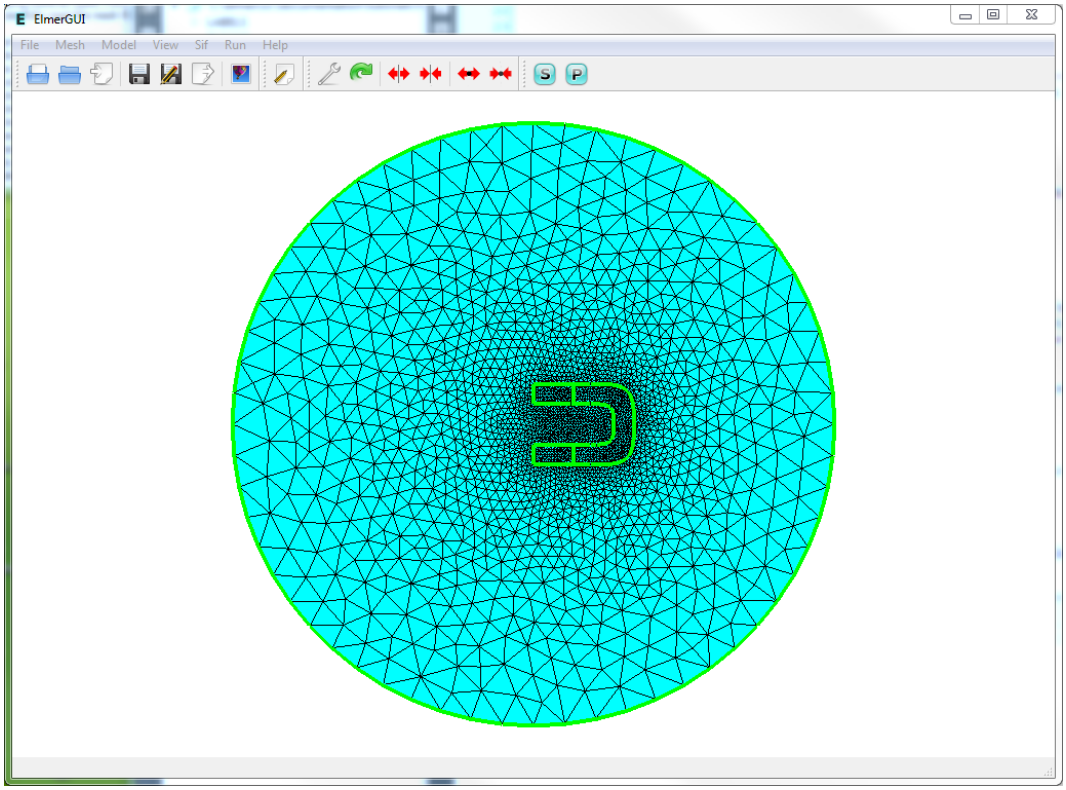

Figure 8.1: The mesh for the horseshoe and surrounding air as see in ElmerGUI

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 the to 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 the 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 differenyt 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 Materail 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 farfield 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 nonlinear 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 8.2. The corresponding postprocessed magnetic field intensity is depicted in Figure 8.3. Note that the derived fields is enforced to be continuous by default which is not optimal for visualization. For optimal results use Discontinous 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.

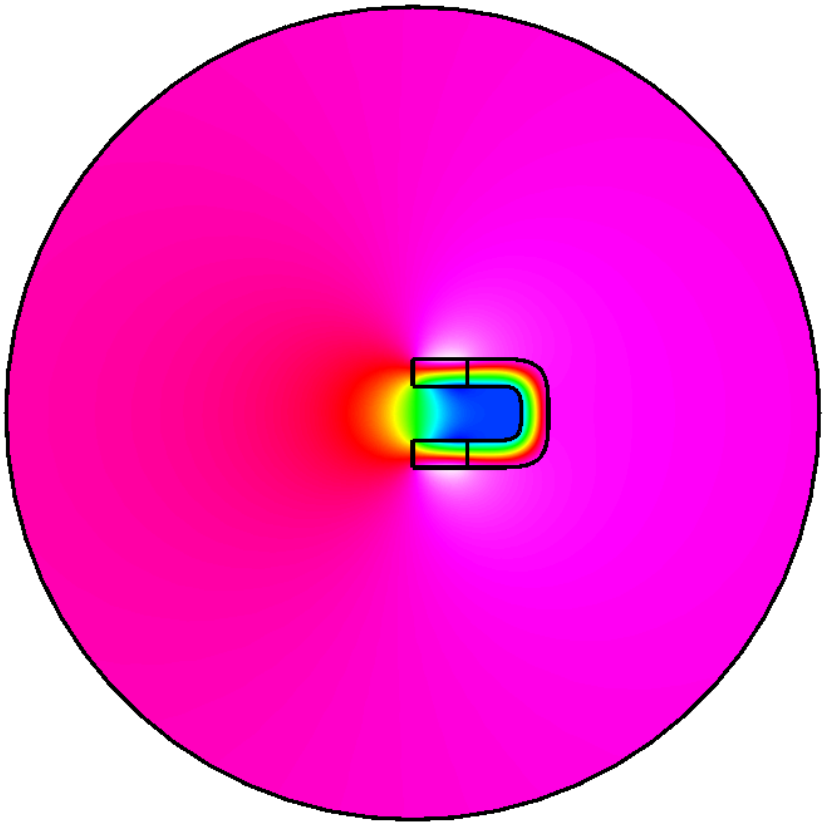

Figure 8.2: The vector potential of the magnetic field.

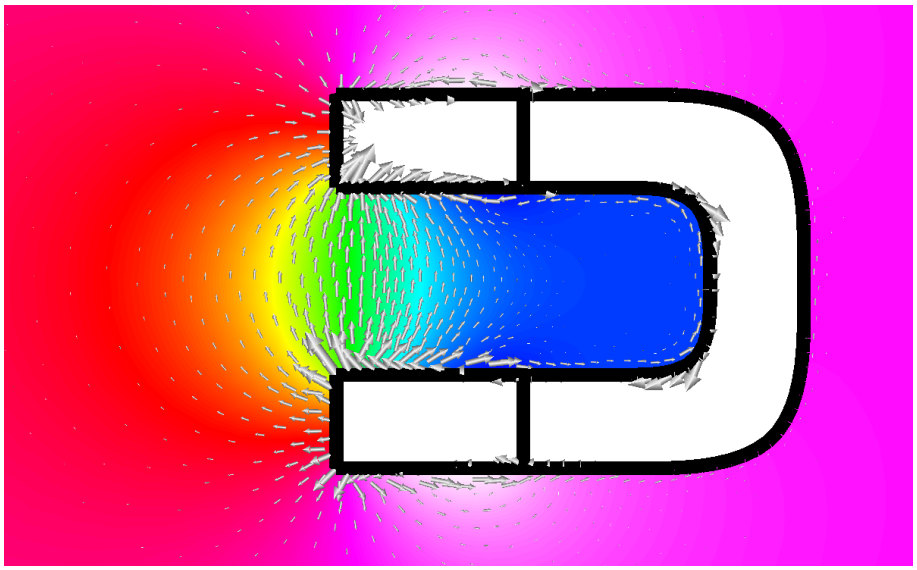

Figure 8.3: A closeup of the vector potential combined with the magnetic field intensity vectors. Note that the fields in the horseshoe itself have been masked away to demonstrate the well known field shape in the free space.

Tutorial 9

Navier-Stokes equation – Laminar incompressible flow passing a step

Directory: FlowStepGUI
Solvers: FlowSolve
Tools: ElmerGUI
Dimensions: 2D, Steady-state

Case definition

This tutorial represents the canonical step flow of viscous fluid. A fluid, flowing past a step (see figure 9.1), has the density 1 kg/m and viscosity 0.01 kg/ms. The velocity profile at the inlet is parabolic with a mean velocity $< v_x >= 1.0$ 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 9.1: Geometry of the step flow problem

Mathematically the problem to be solved is

$$\begin{cases} -\nabla \cdot (2\mu\bar{\bar{\varepsilon}}) + \rho\vec{u} \cdot \nabla\vec{u} + \nabla p &= 0 \quad \text{in } \Omega \\ \nabla \cdot \vec{u} &= 0 \quad \text{in } \Omega \end{cases} \tag{9.1}$$

with the boundary conditions

$$\begin{cases} u_x &= 1 \quad \text{on } \Gamma_{inlet} \\ u_x &= 0 \quad \text{on } \Gamma_{no-slip} \\ u_y &= 0 \quad \text{on } \Gamma_{inlet} \cup \Gamma_{outlet} \cup \Gamma_{no-slip} \end{cases} \tag{9.2}$$

where μ is the viscosity, $\bar{\bar{\varepsilon}}$ 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 the to 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 nonlinear 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 to 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 9.2 and 9.3. 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.

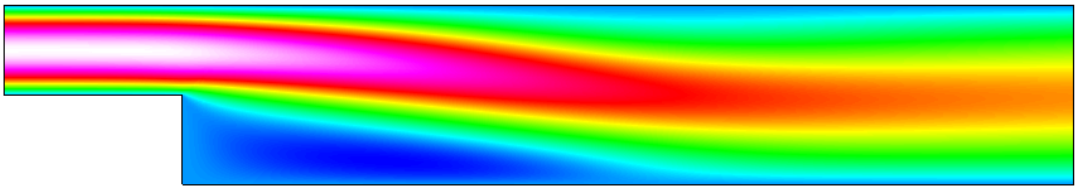

Figure 9.2: Absolute value of the velocity field

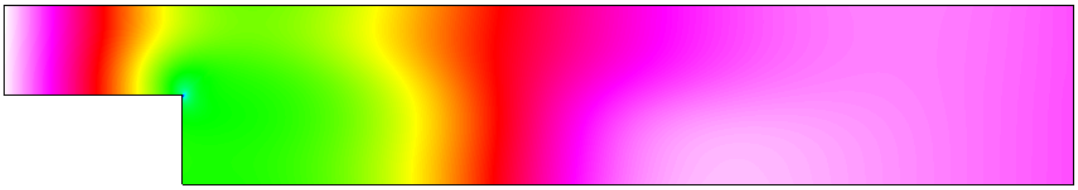

Figure 9.3: Pressure field

Extra task: Decreasing the viscosity

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

- Introducting a relaxation factor (typically in the range 0.5–0.7)
- Increasing number of nonlinear 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 10

Vortex shedding – von Karman instability

Directory: VonKarmanGUI
Solvers: FlowSolve
Tools: ElmerGUI
Dimensions: 2D, Transient

Case definition

This tutorial is about simulating the developing of the vortex shedding i.e. the von Karman instability. The geometry is a tube with a circular obstacle. For more details on the problem look at the benckmark case definition by by M. Schäfer and S. Turek in *"Benchmark computations of laminar flow around a cylinder"*.

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. This is a rather sparse mesh. To increase the element number

```
Mesh
 Configure
   nglib / Max H: 0.02
Mesh
  Remesh
```

This mesh includes 3464 nodes and 6506 triangles. The mesh is presented 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. 2nd order bdf time-stepping method is selected with 200 steps and we want the total simulation time to be 8 seconds.

```
Model
  Setup
    Simulation Type = Transient
    Steady state max. iter = 1
    Time Stepping Method = bdf
```

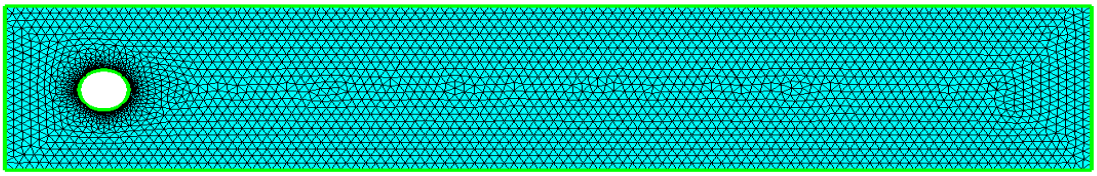

Figure 10.1: Computational mesh of the problem.

```
    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 explicitely 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.

```
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 timestep 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 Paraiew as the time-steps are computed, or wait until all timesteps have been computed. You need to reload the files if number of timesteps are increased.

In Figure 10.2 the velocity field is presented for three different timesteps. 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.

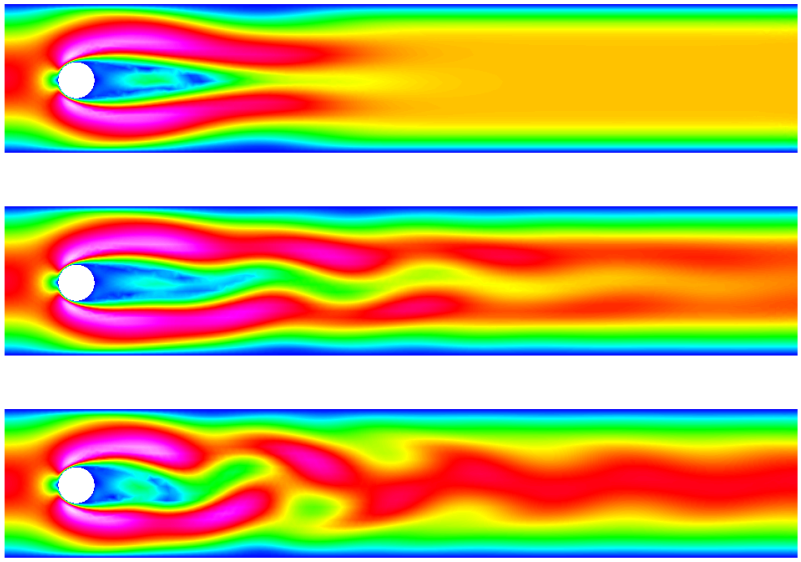

Figure 10.2: Velocity distribution at steps 20, 100 and 200

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 vortecis even more pronounced until they break into fully chaotic motion. This finite element mesh will allow only minor increase in Reynolds number to be able to capture the phenomena.

Tutorial 11

Thermal flow in curved pipe

Files: curved_pipe.grd
Solvers: HeatSolve, FlowSolve
Tools: ElmerGUI
Dimensions: 3D, Steady-state

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 bend 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 and 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 100 the unsteady nature of the flow should probably 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
```

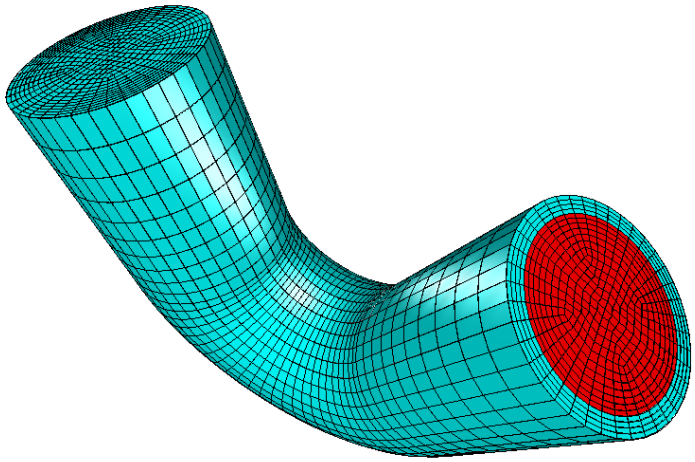

Figure 11.1: The mesh of the curved pipe as seen in ElmerGUI

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

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 the to 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 proconditioner `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 trough the material parameters of the various solvers to ensure that the properties are indeed as they should be. Any consistant 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
```

Unfortunatly 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 know 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.

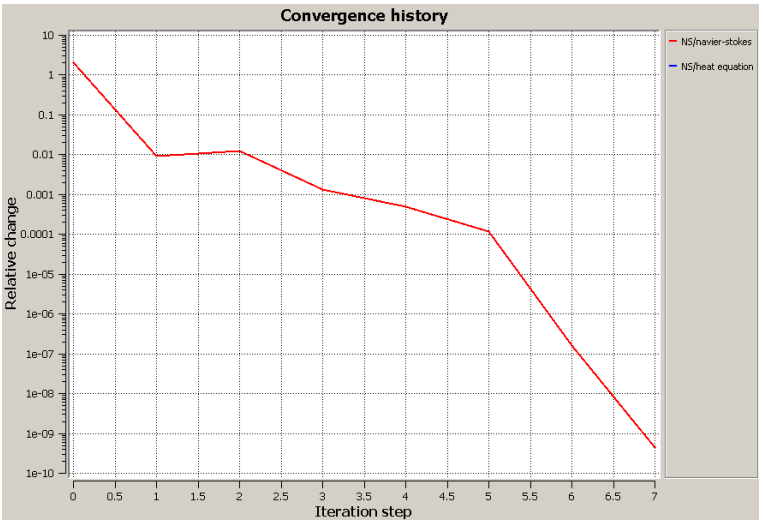

Figure 11.2: Convergence history of the case

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
```

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 crosssection contains any information. It may be seen in Figure 11.3 that the symmetry around pipe origin is lost in the bent.

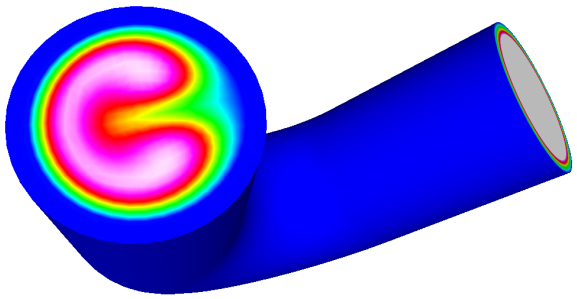

Figure 11.3: Temperature distribution at the outlet of the pipe

Alternatively we may visualize the crosssection 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 11.5 and 11.4 the obtained velocity and temperature distributions are presented.

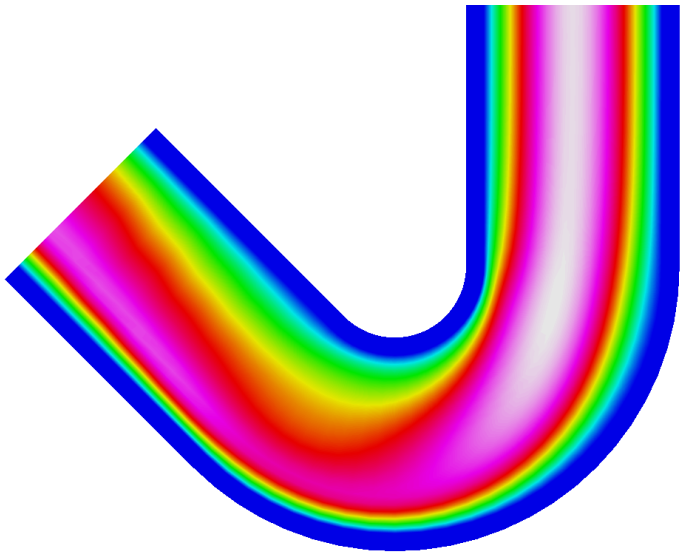

Figure 11.4: Velocity distribution at the cross section $y = 0$.

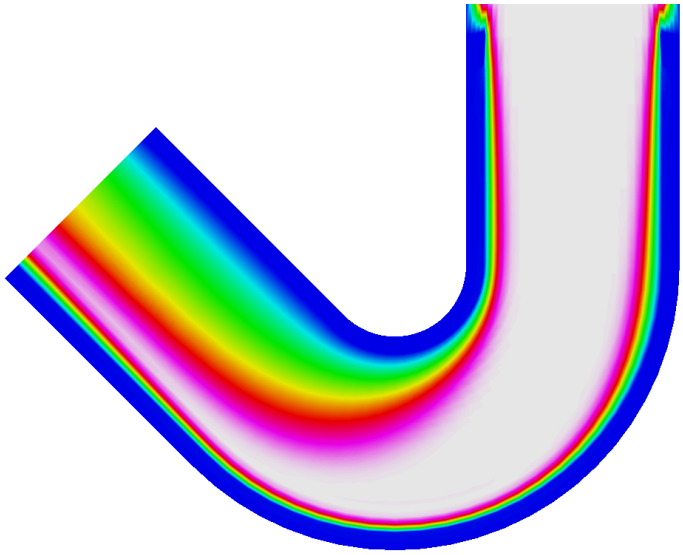

Figure 11.5: Temperature distribution at the cross section $y = 0$.

Tutorial 12

Interaction between fluid flow and elastic obstacle

Files: `obstacle_in_channel.in2d`
Solvers: FlowSolve,ElasticSolve,MeshSolve
Tools: ElmerGUI
Dimensions: 2D, Steady-state

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 fluidic 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, Youngs 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 modeling. 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 nonlinear 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 that it consists of around 4140 nodes and 7890 linear triangles.

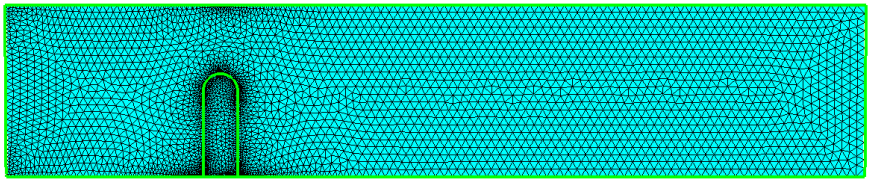

Figure 12.1: The mesh of the obstacle in channel case as seen in ElmerGUI

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 nonlinear 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 nonlinear iterations to 1. The mesh deformation solver just extents 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 as 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 = Elasticty
    Apply to Bodies = 2
    Nonlinear Elasticty
      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 essense of fluid-structure interaction is in the following boundary condition. When the `Fsi BC` is active the fluidic forces are automatically within ElasticSolver. The backcoupling 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 FS
```

For the execution ElmerSolver needs the mesh files and the command file. We have know 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 visulize 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 12.2, 12.3, and 12.4 the obtained velocity, pressure and displacement distributions are presented in the deformed mesh.

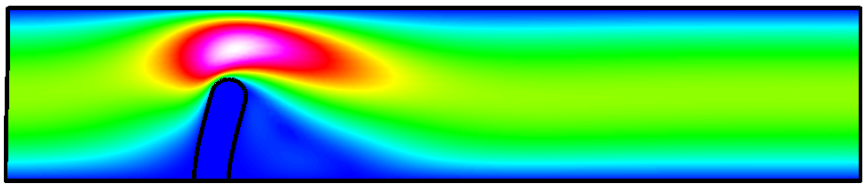

Figure 12.2: Velocity distribution of the obstacle in channel case.

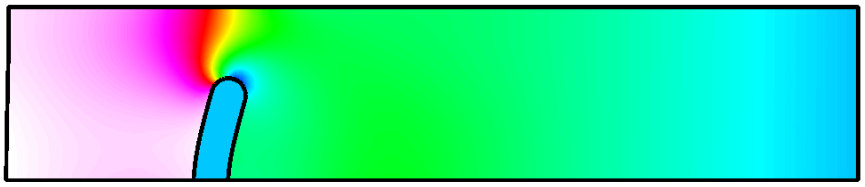

Figure 12.3: Pressure distribution of the obstacle in channel case.

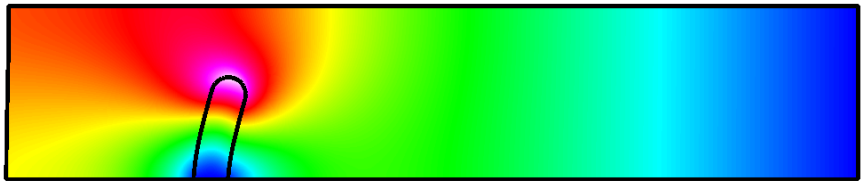

Figure 12.4: Displacement distribution of the obstacle in channel case.

Tutorial 13

Transient flow and heat equations – Rayleigh-Benard instability

Directory: RayleighBenardGUI
Solvers: HeatSolve, FlowSolve
Tools: ElmerGUI
Dimensions: 2D, Transient

Case definition

This tutorial is about simulating the developing of the Rayleigh-Benard instability in a rectangular domain (Figure 13.1) 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 13.1. 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 sate.

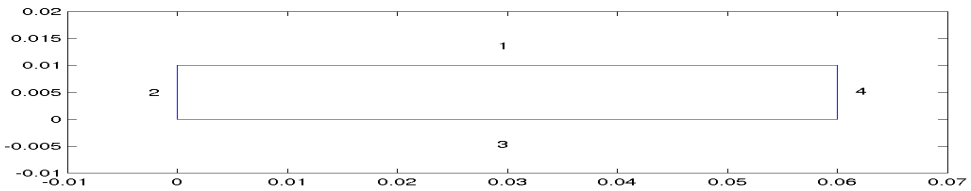

Figure 13.1: Domain.

Table 13.1: 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 that it consists of 3036 bilinear elements.

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 bouyancy force and it is defined by a vector with fourc components. The three 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 the to 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 nonlinear 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 nonlinear iterations in a coupled case low. Here we select just one nonlinear 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 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 water at room temperature from the material library. You may click trough the material parameters of the various solvers to ensure that the properties are indeed as they should be. Any consistant 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 reference temperature for the Boussinesq approximation.

```
Model
  Material
    Material library
      Water (room temperature)
    General
      Reference Temperature = 293
    Apply to Bodies = 1
    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 know 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.

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 timesteps 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 13.2 and 13.3 the obtained temperature distribution and the velocity vectors are presented (note that the pictures are generated by the obsolite ElmerPost software). The maximum velocity in the system should be about 0.516 mm/s.

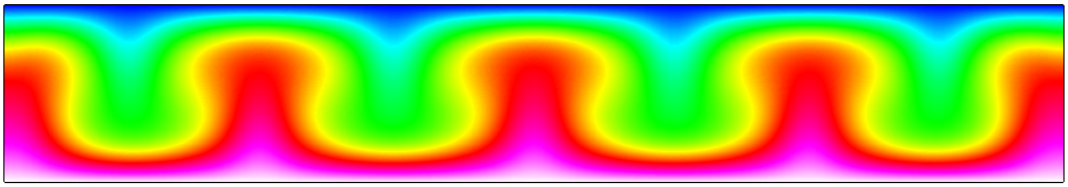

Figure 13.2: Temperature distribution at 260 s.

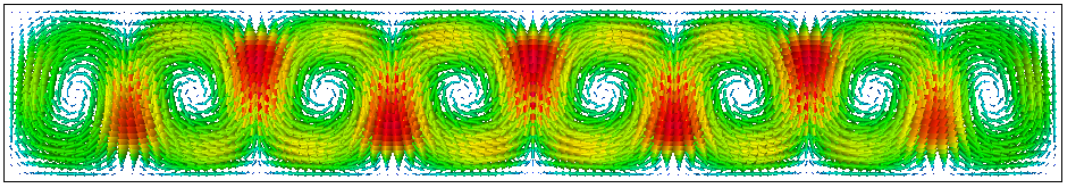

Figure 13.3: Velocity vectors at 260 s.

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 tempereture differences the system eventually becomes steady state and the convection rolls vanish alltogether. 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 14

Electrostatic equation – Capacitance of perforated plate

Files: hexhole.grd
Solvers: StatElecSolver
Tools: ElmerGUI
Dimensions: 3D, Steady-state

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 3×3 mm^2. The holes cover the the other plate uniformly so that the size of each unit cell is 10×10 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

$$C = \varepsilon_r \varepsilon_0 \frac{A}{d}, \tag{14.1}$$

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 $C = 221.36\ fF$.

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 recide 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 is 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.

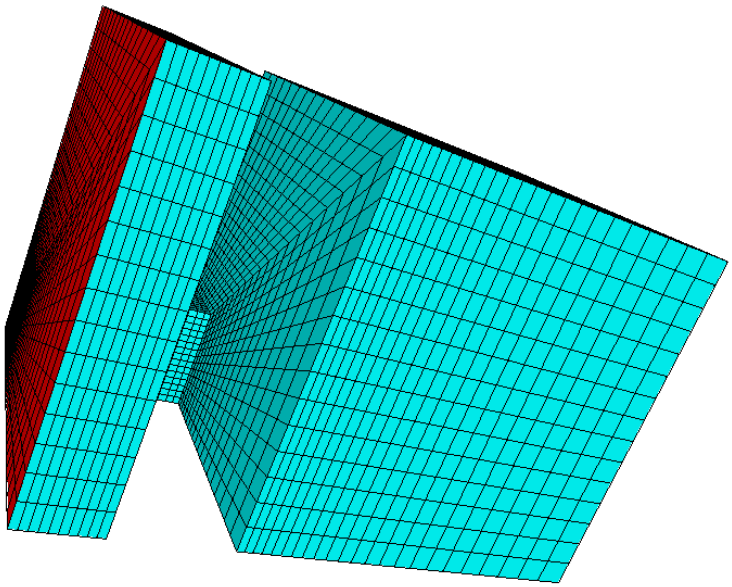

Figure 14.1: The mesh with one highlighted backplate as seen in ElmerGUI

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 the to 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 know 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 nonlinear 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 14.1.

Table 14.1: 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 14.2 some examples of visualizations with paraview are given.

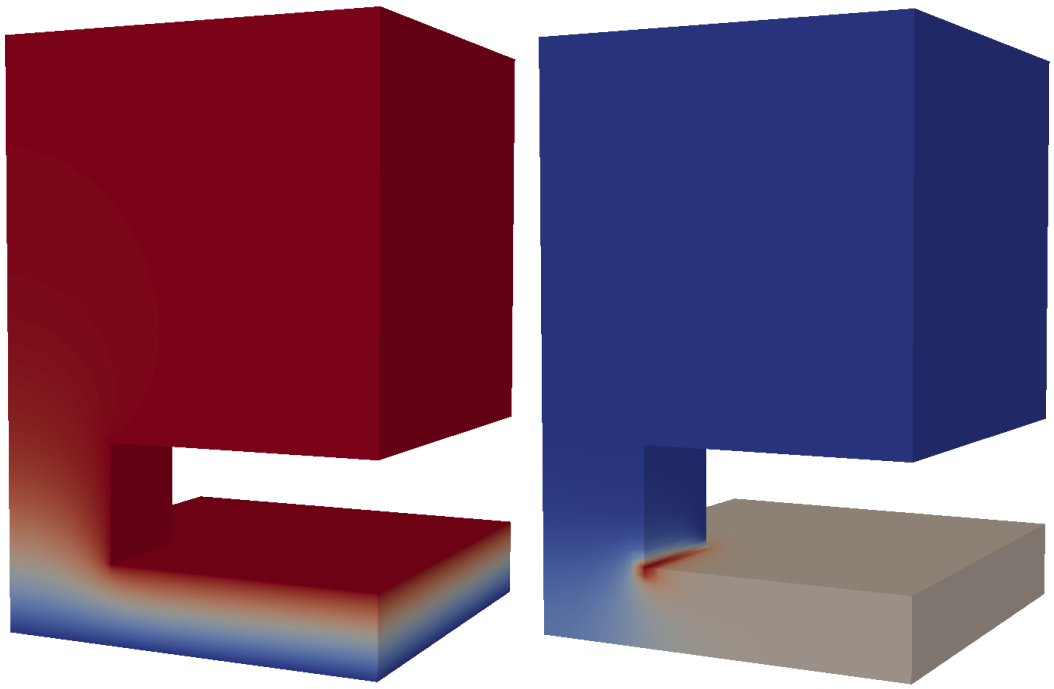

Figure 14.2: The electrostatic potential and the electric energy density of the quarter of a unit cell of a perforated plate capacitor visualized by Paraview.

Tutorial 15

Harmonic magnetic field in 2D - Induction heating of a graphite crucible

Directory: InductionHeatingGUI
Solvers: MagnetoDynamics2DHarmonic
Tools: Gmsh, ElmerGUI
Dimensions: 2D, Axi-Symmetric

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 15.1. The dimensions of the induction heating crucible are given in Table 15.2.

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 1.0e6 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
    Append -> magnetodynamics2d.xml
```

Table 15.1: 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 15.2: 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

The additional definitions should recide 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 recides.

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

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

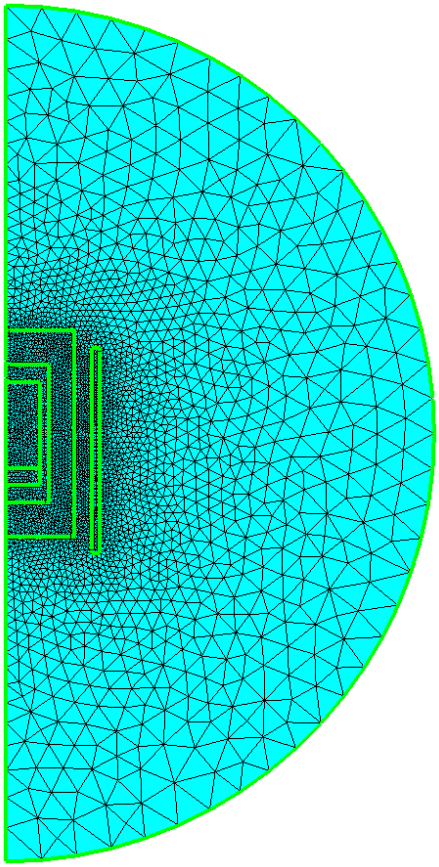

Figure 15.1: The mesh for the horseshoe and surrounding air as see in ElmerGUI

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. 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 the to 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 the 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 differenyt 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 an 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 nontrivial solution.

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

This must be also joinded 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 farfield 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 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.

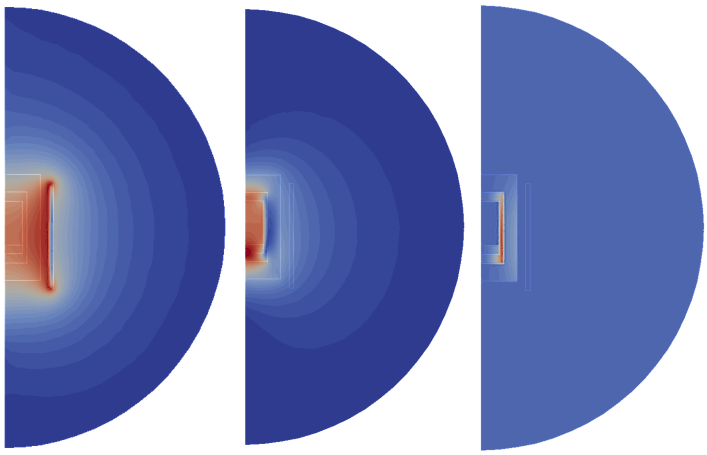

Figure 15.2: Induction heating of a simple crucible. a) in-phase component of the magnetic field intensity b) out-of-phase component of the magnetic field intensity c) Joule losses in the conductors

```
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 nonlinear 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 efficieny is estimated to be 18.9 W. The corresponding results are shown in Fig. 15.2.

It can be noted that would the estimated heating efficieny be different from the known one the user may give the postpressing 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 16

Using VectorHelmholtz module to model wave propagation in bent waveguide

Directory: WaveguideGUI
Solvers: VectorHelmholtz
Tools: ElmerGUI
Dimensions: 3D, Steady-state

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. 16.1.

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.

ElmerGUI solution procedure

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 as follows:

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

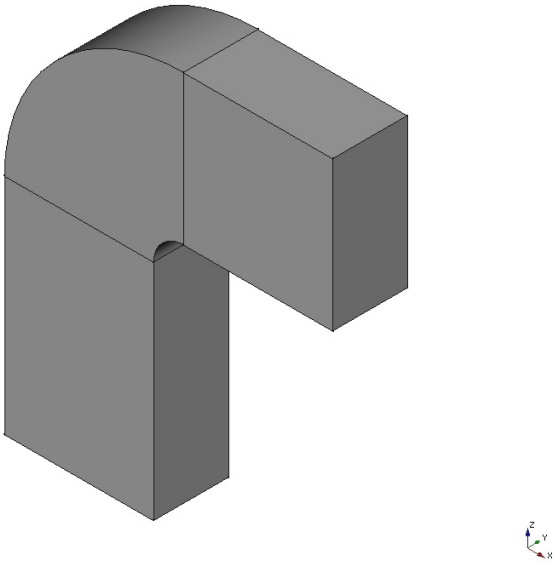

Figure 16.1: Geometry of the waveguide

The `vectorhelmholtz.xml` definition file is, by default, located in `$ELMER_HOME/share/ElmerGUI/edf-extra`.

Load geometry:

```
File
  Open -> waveguide_bend.step
```

As we are modeling a wave phenomenon at $2.5\mathrm{GHz}$ the maximum H in the mesh must be around $\frac{1}{10\lambda}$, where $\lambda \approx 8.3$:

```
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
    Vector Helmholtz Equation
      Active = on
      Apply to Bodies: Body 1
      Angular Frequency = $ 2*pi*2.5e9
      Priority 5
      Free text input
        $ a = 10e-2
```

```
      $ b = 5e-2
      $ c0 = 1/sqrt(8.854e-12*4*pi*10^-7)
      $ omega=2*pi*2.5e9
      $ k0 = omega/c0
      $ kc = pi/a
      $ beta0 = sqrt(k0^2-kc^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
  Inverse Relative Permeability = 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 (prepended 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. 16.2 the Poynting vector and the real part of the electric field of the solution field is shown.

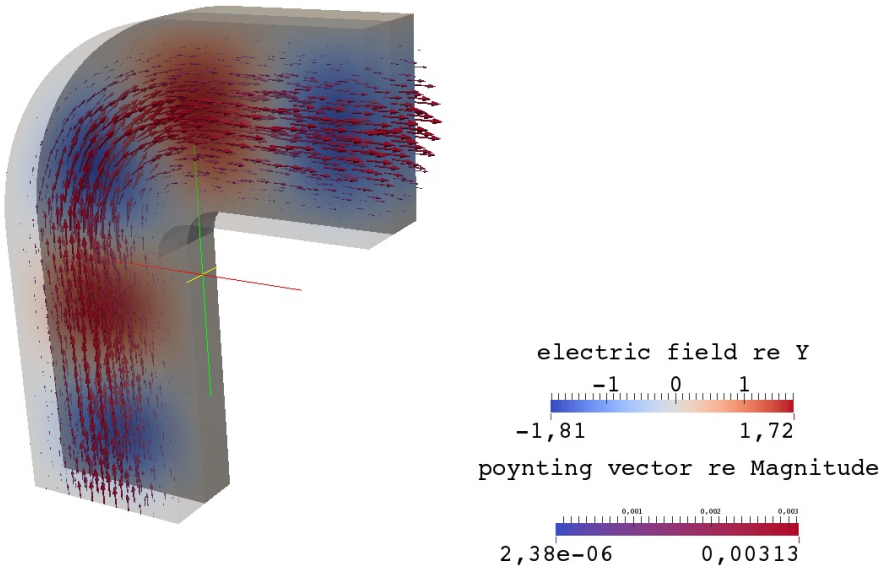

Figure 16.2: The resulting field solution: The y -component of the real part of the electric field and the real part of Poynting's vector

Tutorial 17

Temperature distribution of a toy glacier

Directory: ToyGlacierTemperature
Solvers: HeatSolver
Tools: ElmerGUI,nglib
Dimensions: 2D, Steady-state

Introduction

The purpose of this simple tutorial is to be an introduction into Elmer for people dealing with computational glaceology. 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 $q = 0.02$ W/m^2 is applied at the bottom of the glacier while the surface stays at a fixed temperature of $T_0 = -10$ C. The material properties of ice are used for the heat conductivity $\kappa(T)$. The temperature distribution in the glacier may be solved from

$$\begin{cases} -\kappa\Delta T &= 0 & \text{in } \Omega \\ T &= T_0 & \text{on } \Gamma_D \\ \kappa\frac{\partial T}{\partial n} &= q & \text{on } \Gamma_N \end{cases} \tag{17.1}$$

Starting and meshing

Start `ElmerGUI` from command line or by clicking the icon in your desktop (or in the /bin directory of you 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_toy.in2d` in the samples directory of ElmerGUI, load this file.

Figure 17.1: 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 `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 the to 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 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. 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.

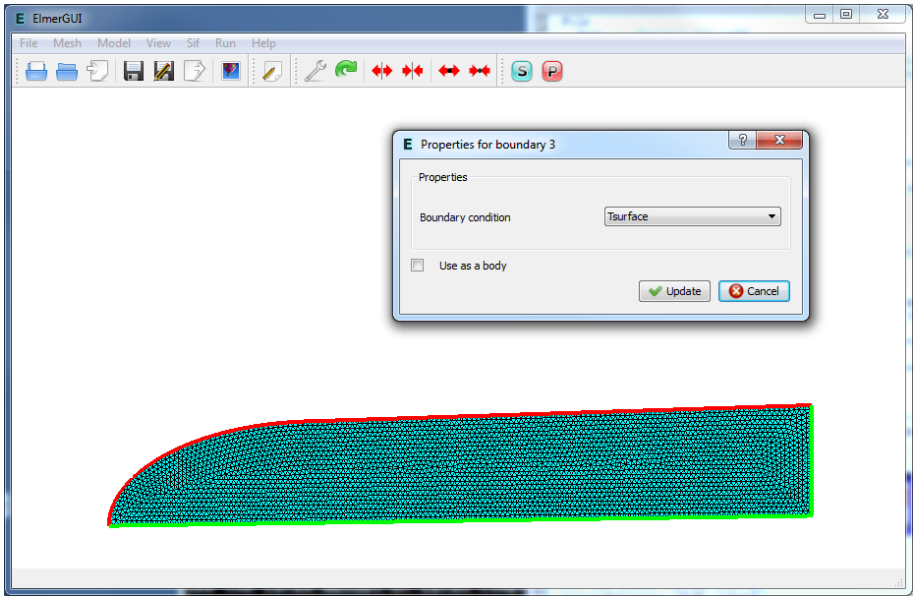

Figure 17.2: Defining boundary conditions in ElmerGUI session

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 17.2.

```
Model
  BoundaryCondition
    Add
      Heat Equation
        Temperature = -10.0
      Name = Tsurface
      OK
    Add
      Heat Equation
        Heat Flux = 0.02
      Name = Tflux
      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 know basically defined all the information for ElmerGUI to write the command file. After writing it we may also visually inspect the command file.

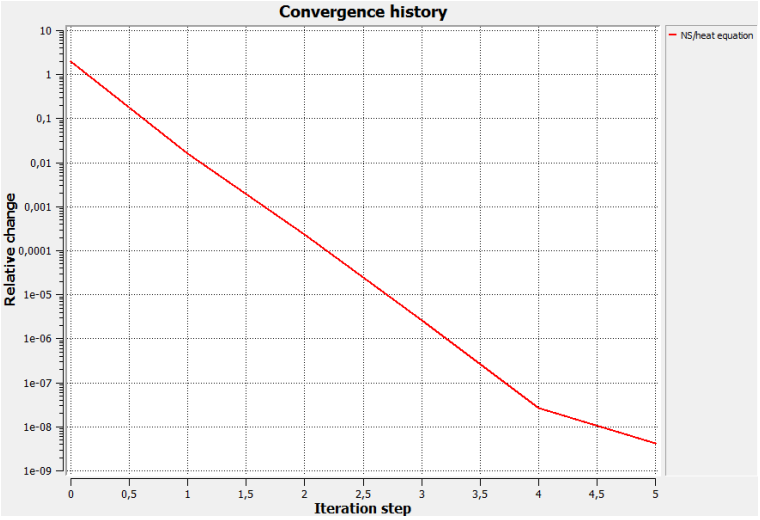

Figure 17.3: The convergence ElmerGUI

```
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 nonlinear iteration. The resulting output is shown in figure 17.3.

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 17.4 shows the surface mesh colored with temperature. Note that these results were carried out with the obsolite 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.

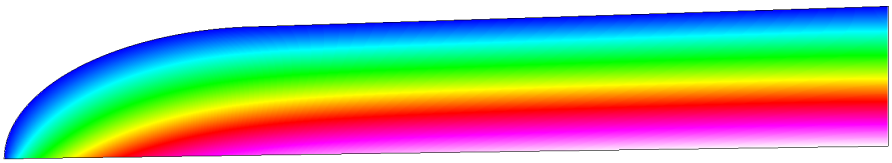

Figure 17.4: The temperature distribution of the toy glacier.

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 consistant with the boundary condition at the top wall.

```
Model
  Initial Condition
    Name = Initial Guess
    Heat Equation
      Temperature = -10.0
```

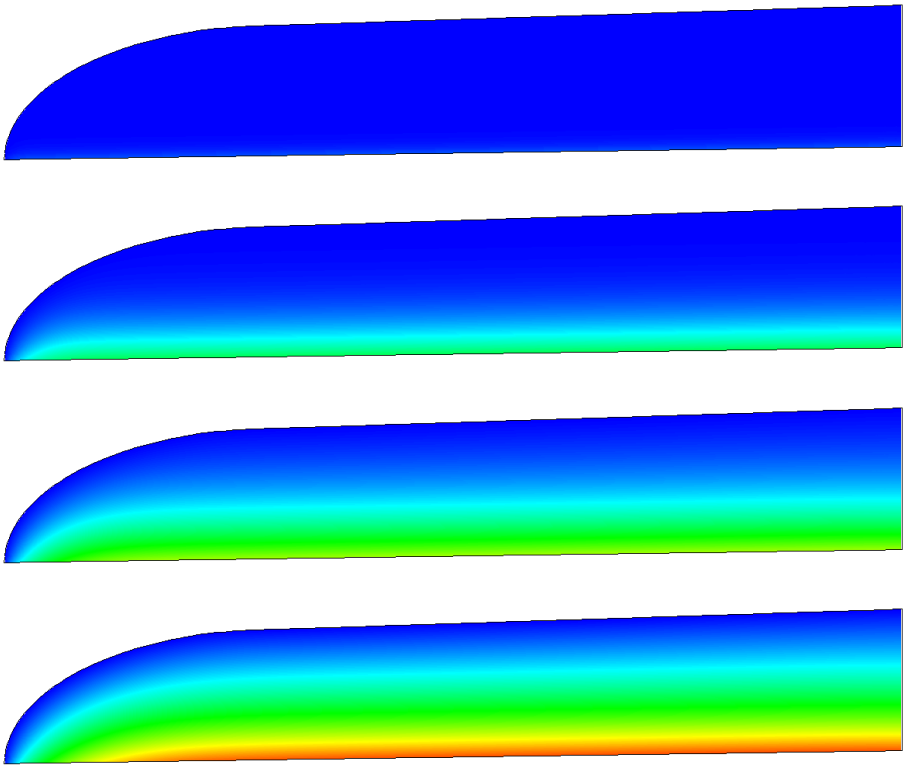

Figure 17.5: Temperature distribution after 1, 20, 50 and 100 timesteps. The temperature scale is the same that is used in the steady-state case. The maximum temperature at end should be about -3.7611 C.

Tutorial 18

Temperature and velocity distributions of a toy glacier and bedrock

Directory: ToyGlacierTemperatureAndFlow
Solvers: HeatSolver,FlowSolver
Tools: ElmerGUI,nglib
Dimensions: 2D, Steady-state

Introduction

The purpose of this simple tutorial is to be an introduction into Elmer for people dealing with computational glaceology. 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 an 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 $q = 0.02$ W/m^2 is applied at the bottom of the bedrock while the surface stays at a fixed temperature of $T_0 = -10$ 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 you 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 18.1.

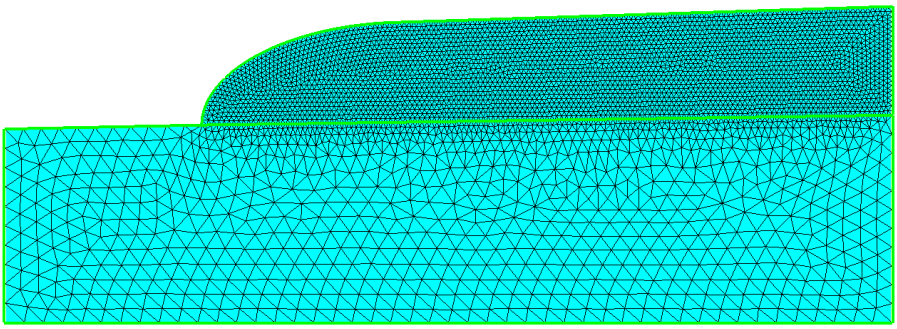

Figure 18.1: The finite element mesh in ElmerGUI

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 the to 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 Celcius) 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
      OK
    Add
      Heat Equation
        Heat Flux = 0.02
      Name = Tflux
      OK
    Add
      Navier-Stokes
        No-slip Wall BC = on
      Name = NoSlip
      OK
    Add
      Navier-Stokes
        Velocity 1 = 0.0
      Name = Symmetry
      OK
```

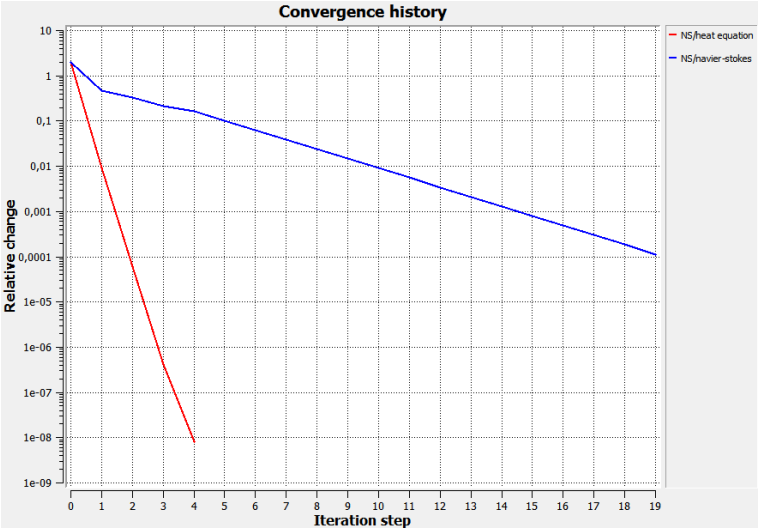

Figure 18.2: The convergence ElmerGUI

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 know 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 nonlinear iteration. The resulting output is shown in figure 18.2.

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

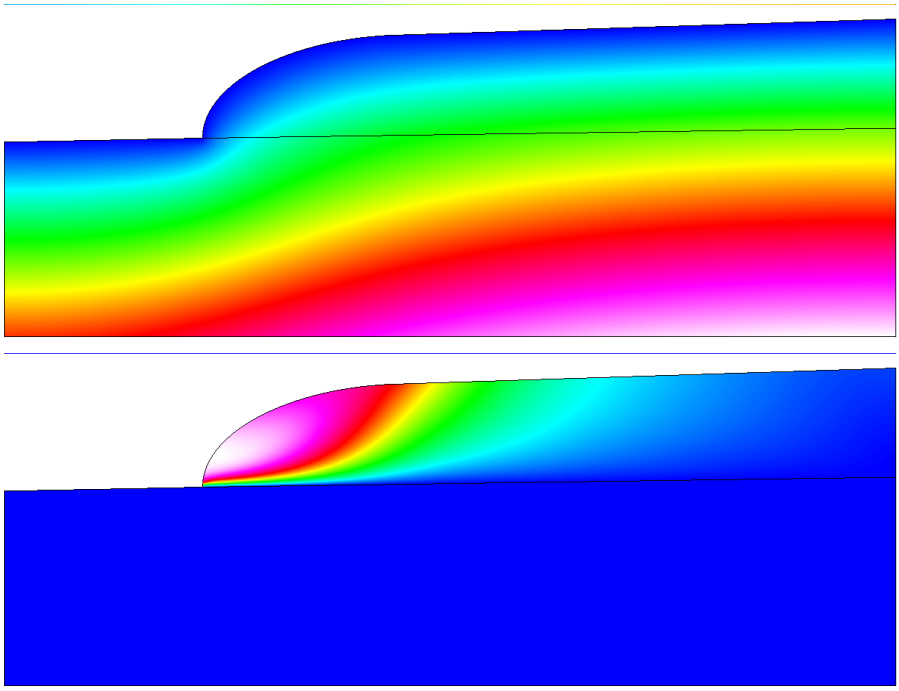

Figure 18.3: Temperature (upper figure) and velocity (lower figure) distributions of the toy glacier sitting on a bedrock.

Results

To view the results we use Paraview,

```
Run
  Start Paraview
```

The resulting temperature and velocity distributions are shown in figure 18.3 (these are generated with the obsolite 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.