Elmer Tutorials

CSC – IT Center for Science

May 24, 2010

Copyrights

About Elmer Tutorials

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

The tutorials starts with problems which require the use of ElmerGUI, the graphical user interface. However, also problems which assume only the use of an text editor are given. There are also obsolite problems that utilize the old graphical user interface, ElmerFront. These are provided only for backward compability but should rather not be studied by new users.

The present manual corresponds to Elmer software version 6.0. Latest documentations and program versions of Elmer are available (or links are provided) at `http://www.csc.fi/elmer`.

Contents

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

$$\left\{ \begin{array}{cccc} -\kappa\Delta T & = & \rho f & \text{in} \quad \Omega \\ T & = & 0 & \text{on} \quad \Gamma \end{array} \right. \tag{1.1}$$

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.

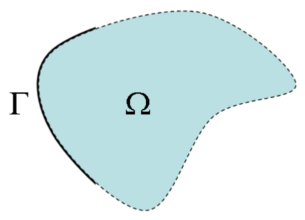

Figure 1.1: Generic object being heated

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.

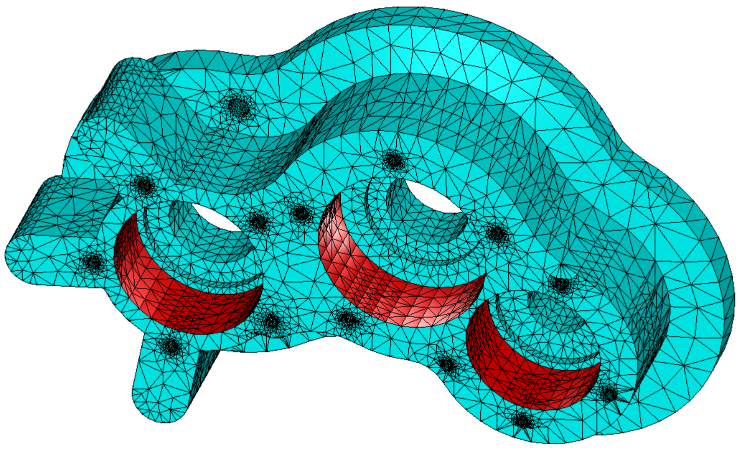
Figure 1.2: The computational mesh showing the three joined boundaries

```
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 may use the ElmerPost postprocessor or start the the internal VTK widget as is done here,

```
Run
  Postprocessor (VTK)
```

The default configuration shows just the object. To color the surface with the temperature choose

```
Surfaces
  Surface: Temperature
  Apply
```

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.

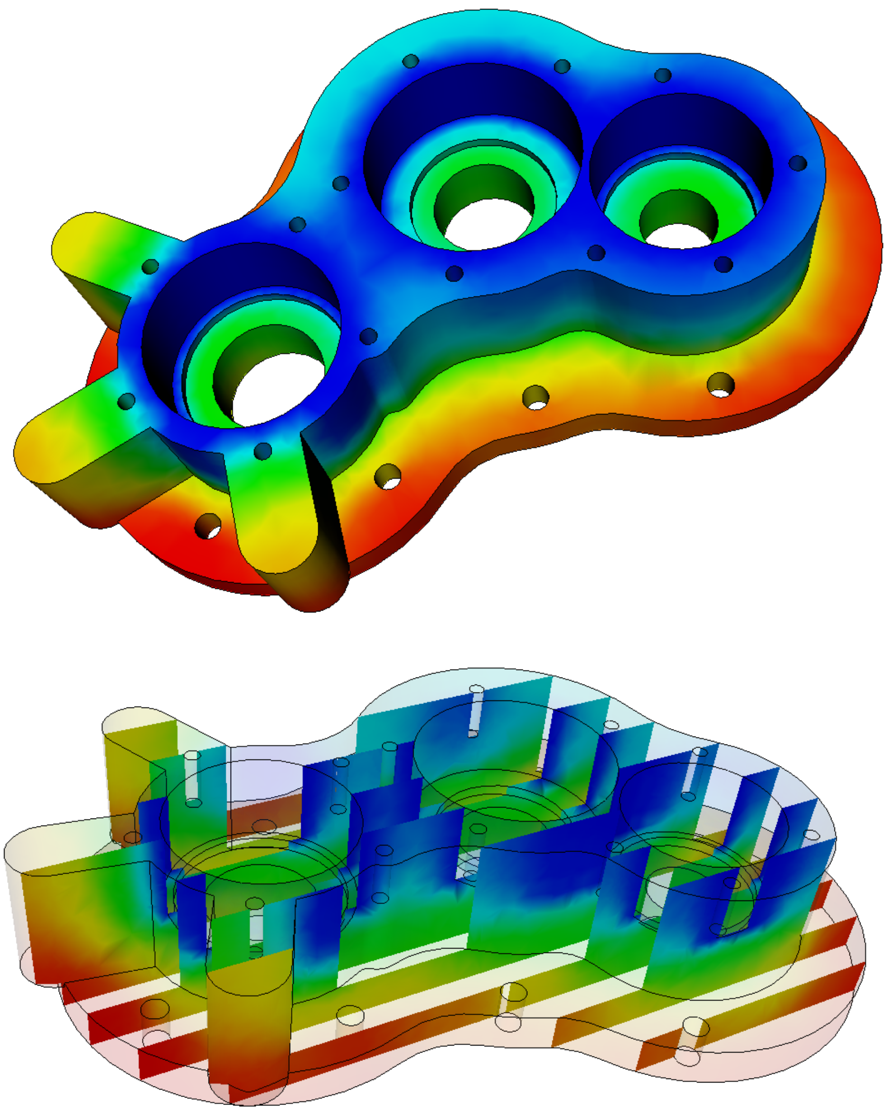
Figure 1.3: The temperature distribution of the solid object domain as visualized using the VTK-based postprocessor

Tutorial 2

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.

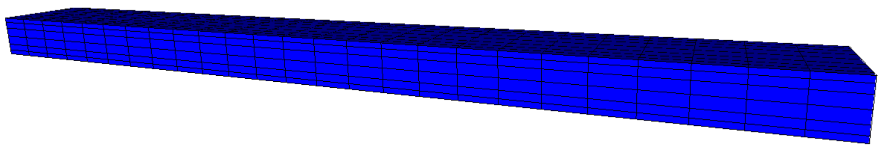

Figure 2.1: The mesh used in the computations

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 ElmerPost.

```
Run
  Start postprocessor
```

As a result the absolute value of maximum displacement is shown. The maximum displacement is 6.36 cm To visualize the displacement in the geometry using ElmerPost can be done with the following command in the `Elmer-Post` command line.

```
  math n0=nodes
  math nodes=n0+Displacement
```

To redraw the picture with new settings use the rightenmost icon on the top row. The resulting picture is shown in Fig 2.2 Note that the displacement are so large that the assumption of linearity may be severely

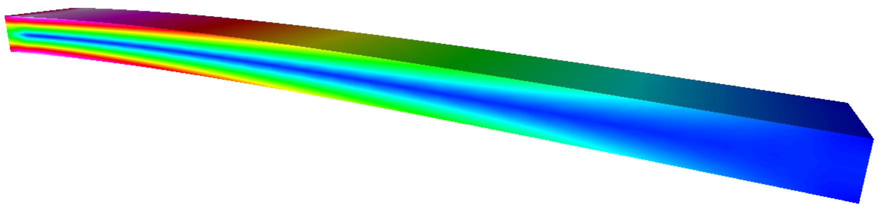

Figure 2.2: The displaced shape of the elastic beam colored with the von Mises stresses

questioned. When further increasing the loading one should resort to a solver that is able to catch the geometric nonlinearities.

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 3

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
```

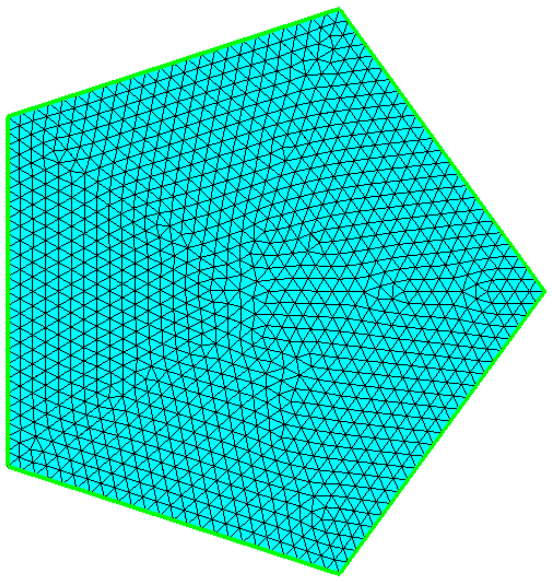

Figure 3.1: The finite element mesh in ElmerGUI

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 3.1.

After we have the mesh we start to go through the Model menu from the top to bottom. In the `Setup` we choose things related to the whole simulation such as file names, time stepping, constants etc. The simulation is carried out in 2-dimensional cartesian coordinates and in 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 3.1. 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 3.1: 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 may start the ElmerPost or use the internal VTK widget, as is done here

```
Run
  Postprocessor (VTK)
```

To show the 1st component

```
  Surfaces
    Control / Surface: Deflection.1
    Apply
    OK
```

The default configuration shows only the 1st eigenmode. To get all the eigenmodes do the following:

```
  File
    Read input file
      Timesteps / End: 10
      Apply
      OK
```

To go through all eigenmodes (treated here as timesteps)

```
  Edit
    Time step control
      Loop
```

Here you may also save the pictures to files frame*.png by activating the checkbox. In figure 3.2 the lowest eigenmodes are depicted.

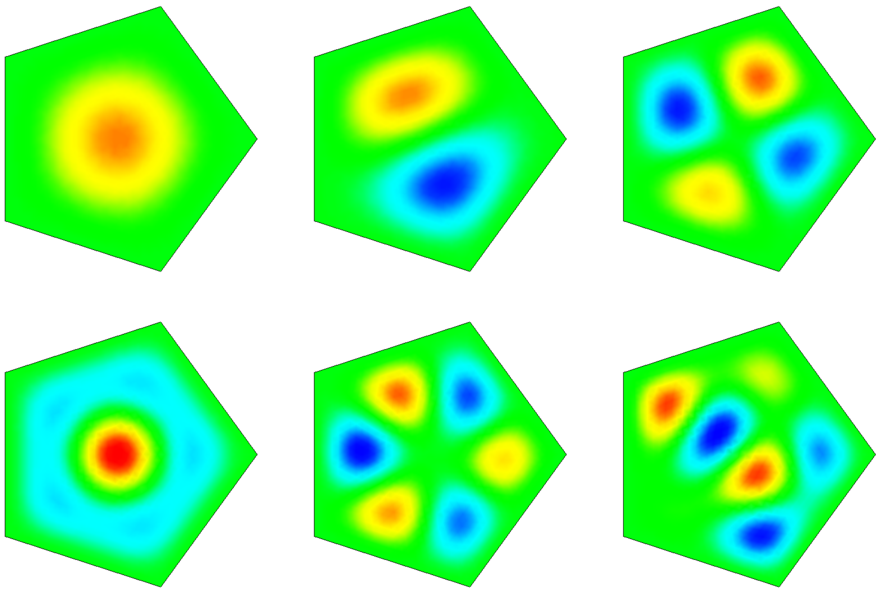

Figure 3.2: The 1st, 2nd, 4th, 6th, 7th and 9th eigenmode of the plate

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.

Tutorial 4

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 4.1), 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.

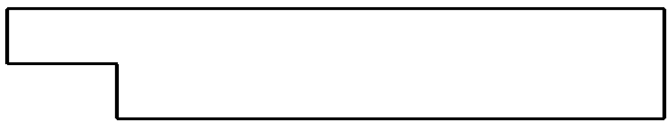

Figure 4.1: Geometry of the step flow problem

Mathematically the problem to be solved is

$$\left\{ \begin{array}{rcll} -\nabla \cdot (2\mu\overline{\overline{\varepsilon}}) + \rho\vec{u} \cdot \nabla\vec{u} + \nabla p & = & 0 & \text{in } \Omega \\ \nabla \cdot \vec{u} & = & 0 & \text{in } \Omega \end{array} \right. \tag{4.1}$$

with the boundary conditions

$$\left\{ \begin{array}{rcll} u_x & = & 1 & \text{on } \Gamma_{inlet} \\ u_x & = & 0 & \text{on } \Gamma_{no-slip} \\ u_y & = & 0 & \text{on } \Gamma_{inlet} \cup \Gamma_{outlet} \cup \Gamma_{no-slip} \end{array} \right. \tag{4.2}$$

where μ is the viscosity, $\overline{\overline{\varepsilon}}$ 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
      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. When there are some results to view we may start the postprocessor also

```
Run
  Start postprocessor
```

Results

The results may be viewed using the postprocessor as shown in Figure 4.2 and 4.3. One may also register specific values, for example the pressure difference is 4.23 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.

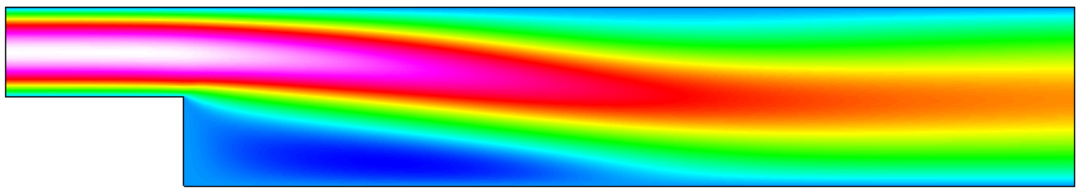

Figure 4.2: Absolute value of the velocity field

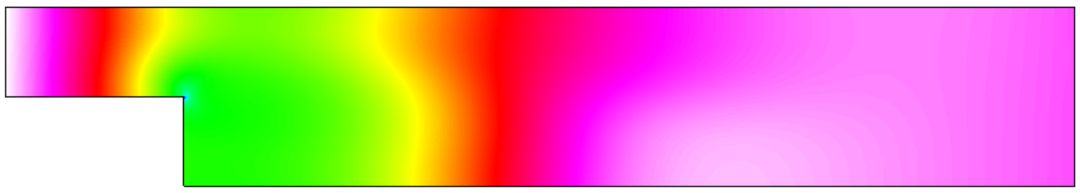

Figure 4.3: Pressure field

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 5

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 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. 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
```

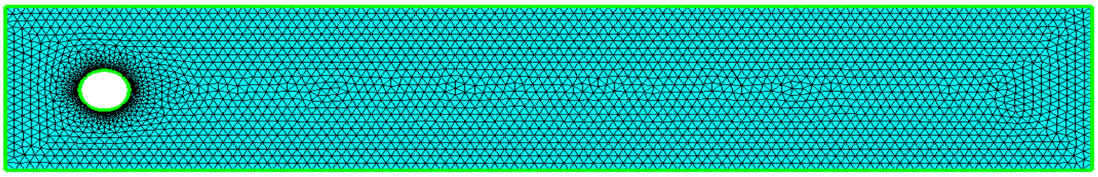

Figure 5.1: Computational mesh of the problem.

```
    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
    General
      Density = 1
    Navier Stokes
      Viscosity = 0.001
    Apply to Bodies = 1
    Add
    OK
```

The system does not need any body forces nor initial conditions i.e. we are happy with the default guess zero.

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 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 postprocessor
```

Results

Due to the number of the time-steps the simulation will take a few minutes. You may inspect the results with ElmerPost as the time-steps are computed, or wait until all timesteps have been computed. When opening the result file using ElmerGUI ElmerPost only opens the first time-step. Therefore it is important to reopen the file and load the time-steps of interest. Pressing the button `All` selects all the calculated time steps. A video of the results can be viewed by selecting the option `Timestep Control` and pressing the button `Loop` under the `Edit` menu.

In Figure 5.2 the velocity field is presented for three different timesteps. The maximum velocity in the system should be about 1.95 m/s.

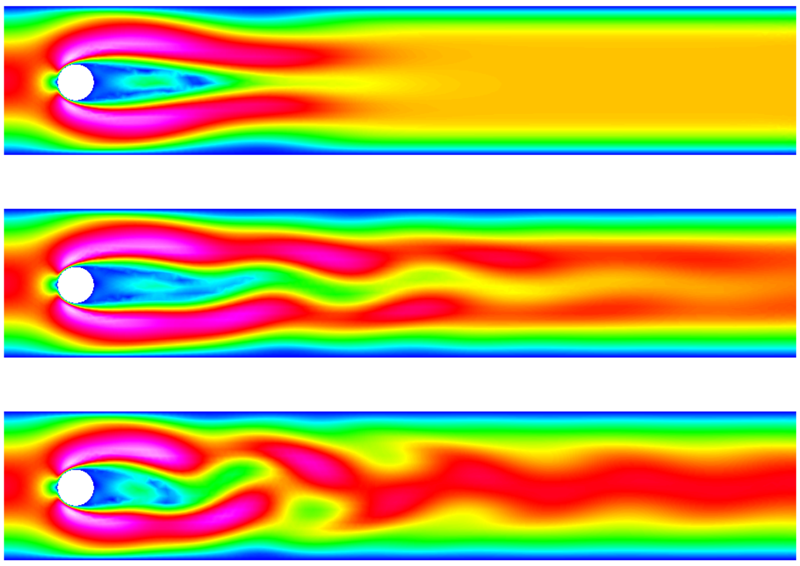
Figure 5.2: Temperature distribution at steps 20, 100 and 200

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 6

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 6.1) 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 6.1. 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.

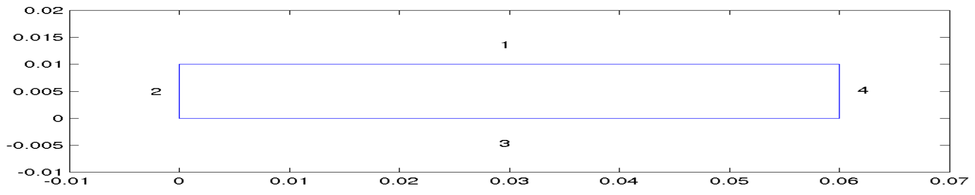

Figure 6.1: Domain.

Table 6.1: 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 `box.grd`, load this file.

```
File
  Open -> box.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.

```
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 = ...
```

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 postprocessor
```

Results

Due to the number of the time-steps the simulation may take around ten minutes. You may inspect the results with ElmerPost as the time-steps are computed, or wait until all timesteps have been computed. When opening the result file using ElmerGUI ElmerPost only opens the first time-step. Therefore it is important to reopen the file and load the time-steps of interest. Pressing the button `All` selects all the calculated time steps. A video of the results can be viewed by selecting the option `Timestep Control` and pressing the button `Loop` under the `Edit` menu.

In Figures 6.2 and 6.3 the obtained temperature distribution and the velocity vectors are presented. The maximum velocity in the system should be about 0.5 mm/s.

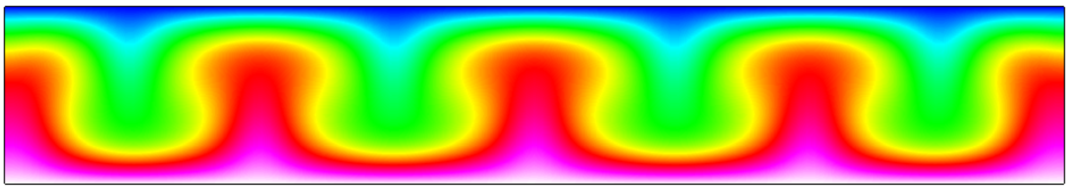

Figure 6.2: Temperature distribution at 260 s.

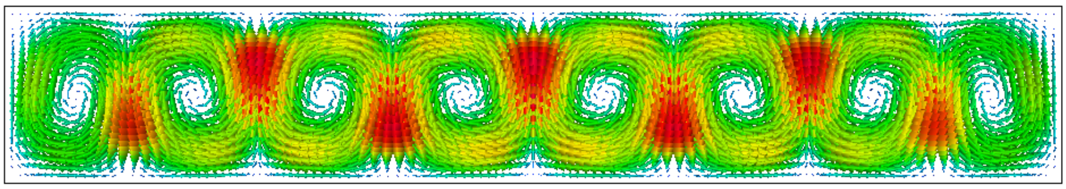

Figure 6.3: Velocity vectors at 260 s.

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 7

Navier-Stokes equation – Turbulent incompressible flow passing a step

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

Case definition

This tutorial is a natural contination of the tutorial 4 where the same case was solved with a smaller Reynolds number. It is advicable to study that case before.

When Reynolds number increases the Navier-Stokes equations do not posses any steady-state solution. Basically the solution can be averaged simulating the transient flow over time. However, the computational cost of this approach is often very heavy particularly while at high Reynolds numbers the computational mesh in direct numerical simulation needs to be very dense. Instead its customary to solve timeaveraged equations. Unfortunately these equations include unknown correlations between quantities that need to modeled in some way.

The workhorse of turbulence modeling is the $k-\varepsilon$ model which is used in this tutorial. The $k-\varepsilon$ model is a two-equation model that introduces two additional variables – the turbulent kinetic energy k and the turbulent dissipation ε which determines the scale of the turbulence.

The case under study is the canonical step flow of viscous fluid. A fluid, flowing past a step has the density 1 kg/m^3 and viscosity $1.0e-4$ kg/ms. The velocity profile at the inlet is defined by a parabolic profile with mean velocity $v_x = 1.0$ m/s and $v_y = 0.0$ m/s. This way the Reynolds number will be 10000. 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.

Also the new turbulent variables require boundary conditions. In the inflow the condition could reflect the values for developed turbulent profile. Here we roughly estimate the turbulent kinetic energy and elongate the distance before the step to have a fully developed turbulent profile. From the literature it is the the turbulent intensity i.e. the kinetic energy of turbulence vs. the kinetic energy of mean flow scalas as $0.16\text{Re}^{-1/8}$. For our current configuration a estimate for the turbulent kinetic energy is 0.00457. For the walls a no-slip condition is applied which also set the values of the turbulent parameters accordingly. Also boundary layer model could be used but here our mesh should be able to capture even the boundary phenomena quite well.

Solution procedure

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

```
File
  Open -> steplong.grd
```

You should obtain your mesh and may check that it consists of 14584 nodes and of 14175 bilinear elements.

```
Model
  Summary...
```

After we have the mesh we start to go through the Model menu from the top to bottom. In the Setup we choose things related to the whole simulation. The steady-state simulation is carried out in 2-dimensional cartesian coordinates, which are also the defaults. The coupled system converges unfortunately quite slowly and hence we need to increase the number of maximum iterations.

```
Model
  Setup
    Simulation Type = Steady state
    Coordinate system = Cartesian
    Steady state max. iter = 100
```

In the equation section we choose the relevant equations and parameters related to their solution. In this case the Navier-Stokes and $k-\varepsilon$ equations are needed. We want to solve the Navier-Stokes and $k-\varepsilon$ equations iteratively using only one nonlinear iteration for optimal convergence of the coupled system. Some relaxation is needed in order to achieve convergence at all. We also relax a little bit on the steady state convergence tolerance. Initially the Navier-Stokes solver uses the more robust Picard iteration which may be changed to Newton iteration after the iteration progresses. However, here we want to supress the use of Newton leariarization since it seems to cause problems with the $k-\varepsilon$ equation.

```
Model
  Equation
    Name = Flow equations
    Apply to Bodies = Body 1
    Navier-Stokes
      Active = on
      Edit Solver Setting
        Nonlinear System
          Max. iterations = 1
          Relaxation factor = 0.5
          Newton after tolerance = 0.0
        Steady state
          Convergence tol. = 1.0e-4
    K-Epsilon
      Active = on
      Edit Solver Setting
        Nonlinear System
          Max. iterations = 1
          Relaxation factor = 0.5
        Steady state
          Convergence tol. = 1.0e-4
    Add
    OK
```

The Material section includes all the material parameters. They are divided 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. For the model parameters of the turbulent equations we are happy with the defaults.

```
Model
  Material
    Name = Ideal
    General
```

```
      Density = 1.0
    Navier-Stokes
      Viscosity = 1.0e-4
      Viscosity Model = K-Epsilon
    Apply to Bodies = Body 1
    Add
    OK
```

The current case does not have any body forces. To help in the convergence we make an rude intial guess.

```
Model
  Initial Condition
    Name = Initial Guess
    Navier-Stokes
      Velocity 1 = 0.0
      Velocity 2 = 0.0
    K-Epsilon
      Kinetic Energy = 0.00457
      Kinetic Dissipation = 1.0e-4
```

When defining Boundary conditions it is possible to assign the to boundaries immediately, or to use mouse selection to assign them later. In this case we have use the latter since we do not necessarily know the numbering of boundaries by heart. There is a special boundary condition that takes care of the Boundary conditions for the noslip walls for both the Navier-Stokes and $k - \varepsilon$ equation. Additionally there are inlet and outlet conditions. For the inlet click `Enter` to open an edit box for the `Velocity 1` when typing in the expression. which will be evaluated at run-time so that $v_x = 6(y-1)(2-y)$.

```
Model
  BoundaryCondition
    Add
    Name = Inlet
    Navier-Stokes
      Velocity 1 = Variable Coordinate 2; Real MATC ``6*(tx-1)*(2-tx)''
      Velocity 2 = 0.0
    K-Epsilon
      Kinetic Energy = 0.00457
      Kinetic Dissipation = 1.0e-4
    Add
    New

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

    Name = Walls
    Navier-Stokes
      Noslip Wall BC = on
    Add
```

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

```
Model
```

```
  Set boundary properties
    Choose inlet -> set boundary condition Inlet
    Choose outlet -> set boundary condition Outlet
    Choose walls -> set boundary condition Walls
```

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

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

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

```
File
  Save Project
```

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

```
Run
  Start solver
```

A convergence view automatically pops up showing relative changes of each iteration. The convergence if far from monotonic but computation should terminate when sufficient convergence is reached after 30 iterations.

Results

When there are some results to view we may start the postprocessor. This time we use the internal VTK based postprocessor for visualization. Also ElmerPost could be used.

```
Run
  Postprocessor (VTK)
```

The results may be viewed using the postprocessor as shown in Figures 7.1 and 7.2. One may also register specific values, for example the pressure difference is 0.302 Pa, the minimum horizontal and vertical velocities are -0.213 m/s and -0.0834 m/s, respectively. One special result of interest is the point, on the x-axis, at which the direction of the flow changes. In this case its position is about 5.1 m after the step.

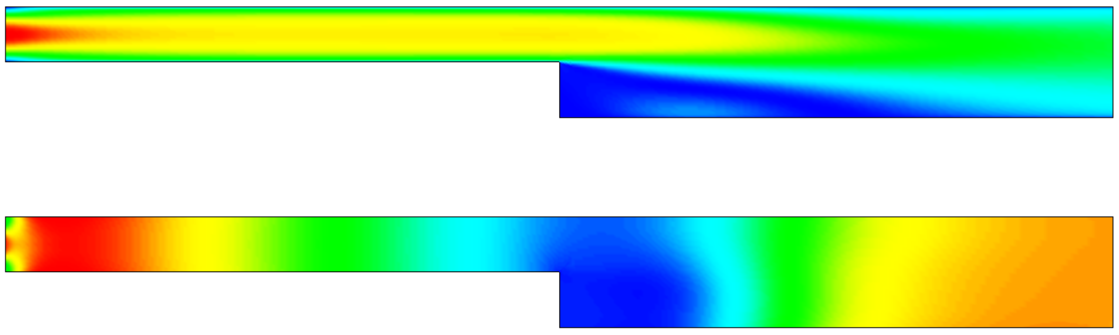
Figure 7.1: Variables of the Navier-Stokes solver: absolute velocity on top and pressure on bottom

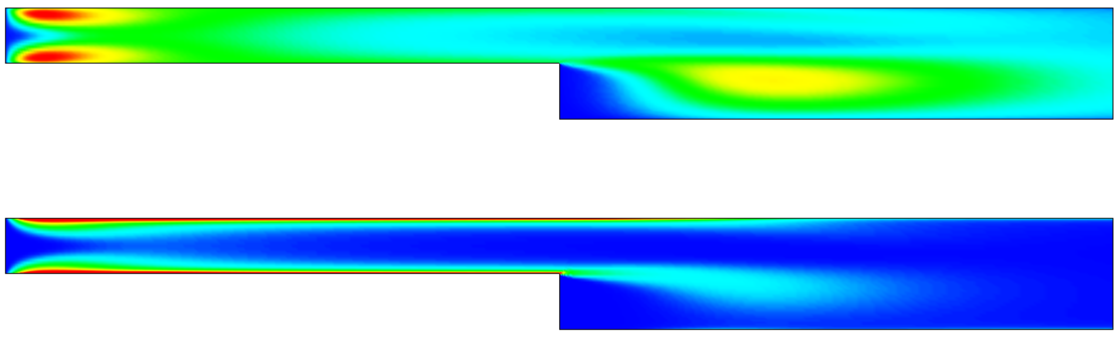
Figure 7.2: Variables of the $k - \varepsilon$ solver: kinetic energy on top and its dissipation on bottom

Tutorial 8

Electrostatic equation – Computation of fringe capacitance

Directory: FringeCapacitance
Solvers: StatElecSolver
Tools: ElmerGUI
Dimensions: 2D, Steady-state

Case definition

This case presents solving the Laplace equation for electric potential.

$$-\nabla \cdot \varepsilon \nabla \phi = 0 \quad \in \ \Omega \tag{8.1}$$

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

$$E = \frac{1}{2} \int \varepsilon |\nabla \phi|^2 \, d\Omega \tag{8.2}$$

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

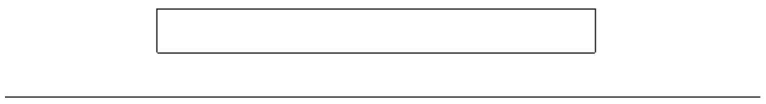

Figure 8.1: The geometry of the capacitor

The geometry studied is a 2D plate capacitor having the well known approximation for Capacitance $C_{app} = \varepsilon_r \varepsilon_0 A/d$. With the help of the simulation one may evaluate the fringe capacitance resulting from the end effects of the capacitor geometry. The measuments of the capacitor are 10×1, and the distance to ground is also 1. Defining the permittivity of vacuum to be $\epsilon_0 = 1$ the comparison to the analytical approximation becomes trivial since then then $C_{app} = 10$.

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

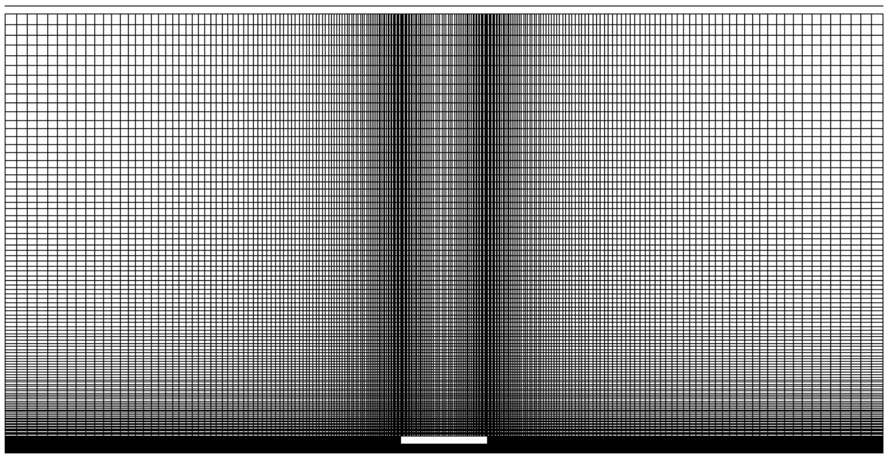
Figure 8.2: Computational mesh used in the simulation

Solution procedure

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

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

The additional definitions should 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 given in ElmerGrid format in file `disc.grd`, load this file.

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

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

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

```
Model
  Setup
    Simulation Type = Steady state
    Permittivity of Vacuum = 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 Electric Field = True
          Calculate Electric Energy = True
    Add
    OK
```

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

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

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

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

    Name = Capacitor
    Electrostatics
      Potential = 1.0
    Add
```

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

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

For the execution ElmerSolver needs the mesh files and the command file. We have 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. When the solution has finished we may start the postprocessor to view some results.

```
Run
  Start postprocessor
```

Results

From the output of the simulation one may see that the capacitance in this case was 13.70 compared to the analytical estimate of 10. Hence the fringe capacitance in this case increases the capacitance by 37 %.

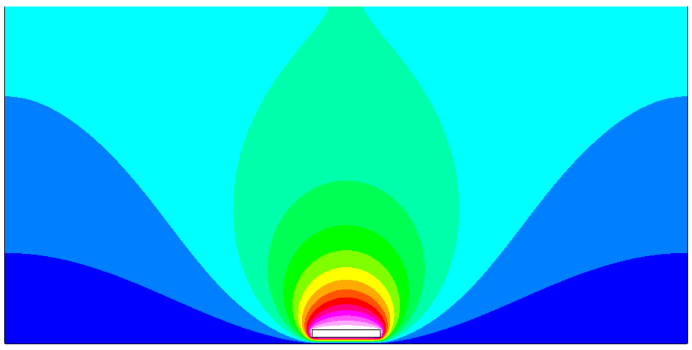

Figure 8.3: The electrostatic potential

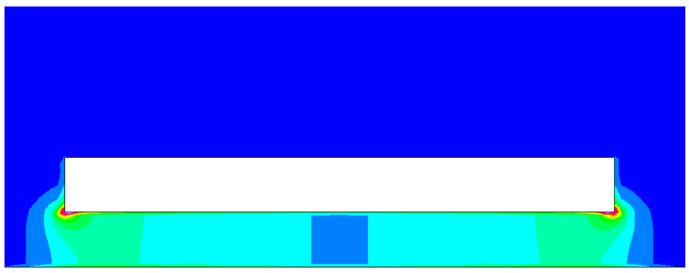

Figure 8.4: The electrostatic energy density, a close-up